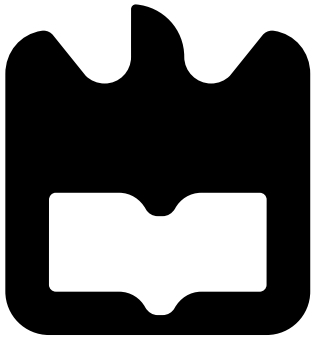
Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática.
2011

Romeu Rabaça Monteiro

CRIAÇÃO E RECONFIGURAÇÃO DE REDES VIRTUAIS NA PERSPECTIVA DO OPERADOR



Romeu Rabaça Monteiro

CRIAÇÃO E RECONFIGURAÇÃO DE REDES VIRTUAIS NA PERSPECTIVA DO OPERADOR

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e Telecomunicações, realizada sob a orientação científica da Professora Dra. Susana Sargento, Professora Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Mestre Jorge Carapinha, Engenheiro do Departamento IEX1 da Portugal Telecom Inovação.

o júri / the jury

presidente / president

Professor Doutor Nuno Miguel Gonçalves Borges de Carvalho
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Pedro Nuno Miranda de Sousa
Professor Auxiliar do Departamento de Informática da Escola de Engenharia da Universidade do Minho

Professora Doutora Susana Isabel Barreto de Miranda Sargento
Professora Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Mestre Jorge Manuel dos Santos Correia Carapinha
Engenheiro do Departamento IEX1 da Portugal Telecom Inovação

agradecimentos /
acknowledgements

Ao longo da realização desta dissertação foram várias as pessoas e entidades que muito me ajudaram e que contribuíram directa ou indirectamente para este trabalho.

Quero agradecer à Prof.ª Doutora Susana Sargento e ao Engenheiro Jorge Carapinha da PT Inovação, que me orientaram ao longo deste trabalho, pela responsabilidade, inteligência, profissionalismo e atitude com que o fizeram.

Aos Mestres Márcio Melo e João Soares quero agradecer todo o apoio que me prestaram, as discussões científicas que tivemos e todos os conselhos que me ofereceram sobre como poderia melhorar o meu trabalho.

Ao Mestre João Nogueira agradeço toda a disponibilidade que teve para me aconselhar e ajudar a resolver os problemas com que me ia deparando, facilitando-me imenso a adaptação a um projecto em progresso.

À Portugal Telecom Inovação pelo acolhimento que me prestaram e pelas condições que me proporcionaram para desempenhar este trabalho, que me beneficiaram bastante tanto a nível profissional como pessoal.

Aos meus pais e ao meu irmão que sempre me apoiaram, incentivaram e deram todas condições para que pudesse ter sucesso, chegar até aqui e ir mais longe ainda.

Ao meu namorado por todo o apoio, incentivo e companheirismo.

E também aos meus vários amigos e amigas que de tantas formas contribuíram para esta dissertação, se interessaram por este trabalho e se preocuparam com o meu sucesso.

A todos e a todas, o meu muito obrigado!

palavras-chave Virtualização, Redes Virtuais, VPN, Cloud, Mapeamento, SAIL

resumo A virtualização é uma peça fundamental no processo actual de inovação em computação e em *networking*. Este conceito de separar as funcionalidades da infraestrutura subjacente permite a consolidação do uso de recursos físicos, reduzindo preços e impacto ambiental. Também permite a conversão de Capital Expenditure (CAPEX) em Operational Expenditure (OPEX), disponibilizando um conjunto de soluções para o mercado de comunicações e computação que é muito interessante para os operadores de telecomunicações. A virtualização de servidores permite que, no mesmo servidor físico, diferentes servidores virtuais possam coexistir partilhando recursos mas permanecendo isolados e independentes, consolidando o uso do servidor físico. Este foi um passo fundamentam para permitir do desenvolvimento de serviços de Cloud Computing (CC).

Apesar de CC ser normalmente definido como "on-demand network access to a shared pool of computing resources" [38], a rede que liga até aos recursos de CC - a Internet - não é controlada pelo cliente. A falta de garantias relativamente à congestão de tráfego e outros aspectos de Quality of Service (QoS) podem inibir empresas que procuram serviços de CC de alta performance e com elevadas garantias de usarem CC. Neste contexto, a virtualização de rede surge como uma oportunidade para ligar todos os recursos através de uma rede virtual gerida pelo cliente e independente do restante tráfego e redes que usam a Internet, com garantias de performance de rede mais do que só de computação.

Assim, esta Dissertação tem como principal objectivo integrar os conceitos de rede e CC e fazer uma gestão integrada de recursos virtuais de rede e cloud. Também se pretende evoluir em duas direcções distintas: Virtual Networks (VNs) & Cloud e Virtual Private Networks (VPNs) & Cloud. Neste sentido, são propostos e estudados mecanismos de mapeamento integrado de recursos de rede e de cloud, assim como soluções para a criação de redes virtuais num modelo semelhante ao das VPNs. Estes mecanismos avançados foram então seleccionados de acordo com os resultados de simulação e integrados na plataforma de virtualização Network Virtualization System Suite (NVSS), a ser desenvolvida no âmbito do projecto SAIL. Posteriormente, foi avaliado experimentalmente o desempenho da plataforma ao nível do mapeamento, em termos da qualidade das soluções de mapeamento e dos tempos envolvidos, tanto nas soluções de VN & Cloud como de VPN & Cloud. Foi também feito um breve estudo experimental sobre a influência da virtualização no tráfego e de que forma ela pode afectar o número máximo de redes virtuais que podem coexistir.

Os resultados obtidos permitem verificar que a NVSS passou a poder gerir recursos cloud para além dos de rede, e a suportar redes em modo VN ou VPN, fazendo uso de um mecanismo de mapeamento integrado de cloud e rede previamente estudado e optimizado.

keywords Virtualization, Virtual Networks, VPN, Cloud, Mapping, SAIL

abstract Virtualization is a key enabler of innovation nowadays both in computing as in networking. This concept of decoupling functionalities from the underlying infrastructure allows for consolidation in the use of physical resources, thus reducing costs and environmental impact. It also allows the conversion of CAPEX to OPEX, making available a set of solutions for the communications and computing market which is very interesting for telecommunications operators. Server virtualization allows that, in the same physical server, several virtual servers can coexist sharing the resources but remaining isolated and independent, consolidating the use of the physical server. This was a fundamental step to enable the development of CC services.

In spite of CC being commonly defined as "on-demand network access to a shared pool of computing resources" [38], the network connecting to the CC resources - the Internet - is not controlled by the client. The lack of guarantees regarding traffic congestion and other QoS aspects might inhibit companies searching for high-performing and highly-guaranteed Cloud computing services from using CC. In this context, network virtualization appears as an opportunity to connect all the resources through a virtual network managed by the client and independent from the rest of the traffic and of other networks using the Internet, with guarantees regarding network performance more than just computing performance.

This way, this Thesis main objective is to integrate the concepts of network and CC and implement an integrated management of cloud and network virtual resources. It is also a goal to evolve into two different directions: VNs & Cloud and VPNs & Cloud. Therefore, mechanisms for the integrated mapping of cloud and network resources were proposed and studied, as well as solutions for the creation of virtual networks in a similar model to that of VPNs. These advanced mechanisms were then selected according to the simulation results and integrated in the NVSS platform, which is being developed within the SAIL project. Afterwards, the platform's mapping performance was experimentally evaluated, regarding the quality of mapping solutions and computing times, both in the solutions for VN & Cloud as for VPN & Cloud. A brief experimental study on the influence of virtualization on traffic and how it can limit the maximum number of coexisting VNs was also conducted.

The results show that the NVSS became able to support cloud resources besides network resources, and also networks in mode VN as well as VPN, while using an integrated mapping mechanism for cloud and network resources, which was previously studied and optimized.

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface

AWS Amazon Web Services

CAPEX Capital Expenditure

CC Cloud Computing

CDN Content Distribution Network

CEAS Cross-Entropy Ant System

CPU Central Processing Unit

CVN Connective Virtual Network

DNS Domain Name System

DRE Distributed Real-time Embedded

FTP File Transfer Protocol

GB Gigabyte

GUI Graphical User Interface

HDD Hard Disk Drive

IP Internet Protocol

ISO International Organization for Standardization

ISP Internet Service Provider

IT Information Technology

LAN Local Area Network

LSP Label-Switched Path

MPLS Multi-protocol Label Switching

NAT Network Address Translation

NR Neighborhood Resource Availability

NV Network Virtualization

NVSS Network Virtualization System Suite

OPEX Operational Expenditure

OS Operative System

PE Provider Edge

QoS Quality of Service

RAM Random-Access Memory

SLA Service Level Agreement

TCP Transmission Control Protocol

UDP User Datagram Protocol

vCPU virtual CPU

VLAN Virtual Local Area Network

VM Virtual Machine

VN Virtual Network

VPN Virtual Private Network

VRF VPN Routing and Forwarding

XML Extensible Markup Language

VMM Virtual Machine Monitor

VPC Virtual Private Cloud

VR Virtual Router

Chapter 1

Introduction & Overview

1.1 Motivation

1.1.1 Moving into the Cloud

"The network is the computer." was the slogan of a marketing campaign launched by Sun Microsystems in the early 1990s. At the time it was an enigmatic slogan, but it is now recognized as a prophecy about the future of computation. From that time to the present, as Internet and computing have become more widespread and available for most of the population, we have entered the age of pervasive computing, when computation and information are available anywhere anytime [37]. We have been using the Internet to connect to our peers, to our universities, governments, to the media and many others, often using their computing and storage resources to our own benefit, thus making the network our computer, or at least an extension of it. We now have e-mail accounts which can store several Gigabytes hosted somewhere unknown to us, but available anywhere and anytime, as long as we're connect to the Internet. The cost savings in providing an e-mail hosting service this way enable operators to offer these services for free, with the only revenues originating in advertising. So, if it's possible to host so many clients and data with such reduced costs to offer e-mail services, why not offer a cheap and pervasive solution for data hosting and computing based in similar premises? This kind of approach to hosting data and providing computation services somewhere in the network was named Cloud Computing (CC), since cloud was the analogy to a general network whose main features are known, but internal details and characteristics are unknown or unavailable.

Nicholas Carr's analogy in "The Big Switch" [15] has been one of the most simple ways to explain CC and to get people enthusiastic about it. Carr makes us look at our modern History, more specifically to the industrial age, when enterprises using mechanical power locally generated through steam engines and waterwheels, and later using local electrical generators, moved their power sources into "the cloud". The cloud here represents the current power generation system, where hyper scaled power generation centrals produce power and then distribute it through a grid to the consumers, allowing for huge costs reductions. This system provides energy in a location-independent, metered, pay-per-use and on-demand model. In the same way, this is foreseen as the future of computation in the form of CC, a service where computing and data storage services are provided by large data centers which distribute them using the network (generally the Internet), with the cost reduction making it an interesting market product. Even though it's a simple and elegant analogy, it should be

considered carefully as Carr himself admits that "all historical models and analogies have their limits" [63]. As of today, there is no parallel in the analogy concerning the data that is passed on to the Cloud providers, which is also a critical element on this revenue and cost equation.

A system like this needs to offer computing and hosting services for each individual client, maintaining the resources being used by each client isolated from one another, in order to safeguard performance and data security. This could be achieved by associating one physical machine to each client, but the savings would be small or none at all, since it would be very similar to each client having its own machine and the use of the network to connect clients and machines would represent an extra cost. Server virtualization is the technology that has started being employed to solve this problem. Using virtualization a number of different virtual servers can coexist in the same physical machine sharing the physical resources while remaining isolated from each other, considering that their data is protected and their performance does not interfere on other virtual servers' performance.

1.1.2 Taking advantage of Network Virtualization

In spite of all these benefits if, for example, an enterprise wishes to outsource some or all of its Information Technology (IT) infrastructure to the cloud, it will need a network connection to connect its headquarters to the place in the cloud where the IT resources are. An Internet connection might be used but this takes the control of the connection to the resources from the enterprise. For example, the connection between the Cloud resources and the enterprise might be experiencing heavy traffic and thus getting slower and performing worse than it would be desirable. Ideally, the enterprise should be able to control the connections from the headquarters to the cloud resources just as it would control if the resources were inside its own network. This is where network virtualization can make a difference and close the gap between the enterprise's physical network and the cloud. By using network virtualization one can create a VN that will perform similarly to a physical network. This is made by instantiating virtual routing nodes and virtual links on the operator's network. Using part of the physical machines just as part of bandwidth of the physical links to create the VN, we can "slice" the physical network of the operator into several VNs. The technologies of virtualization make them isolated from each other, so that each VN can be fully independent and safe from other VNs. By creating and using a VN to connect the enterprise's sites to the cloud resources the enterprise will have outsourced both its network and IT resources, gaining several benefits from this outsourcing while having full control over the network and the IT resources. In the end, it's like creating an extension of the enterprise: it's virtual but the experience is real.

1.1.3 Current Context

While being an appealing solution it has to be developed. Nowadays several enterprises like Amazon[11], Orange[45], Portugal Telecom[49] and many others offer cloud resources and VN solutions. Some like Verizon and IBM [53] offer a package with both network and cloud. But these services are not integrated in any of this cases, even though they might be purchased together. These resources, both virtual and physical, need to be monitored, controlled, created, destroyed and mapped. This means that, when receiving a request to create a VN and cloud resources for a client, the operator must decide which are the physical

machines where the virtual routers will be hosted, the datacenters where the virtual servers will be instantiated and the physical links through which the virtual links will go through. The operator needs to have an up-to-date view of the physical network and of the virtual networks running over it. It needs to have decision mechanisms that will place the virtual machines and virtual links optimally so that the physical infrastructure is efficiently used, by hosting the greatest number of virtual networks as it can possibly take.

Currently there exists a platform, the NVSS, that provides the basic tools to monitor the physical infrastructure of a network and to create, discover, map and manage virtual networks. In spite of this, it only supports network resources and not cloud.

1.2 Purpose

This way, this Thesis aim is to integrate network and cloud concepts, as well as to provide an integrated platform for the management of VN and cloud resources. Regarding the process of mapping virtual networks onto physical networks, this should be done using mapping algorithms which map both network and cloud resources efficiently and in an integrated way.

Thus, a mapping algorithm which maps both network and cloud resources in an integrated approach shall be designed and evaluated, with the goal of maximizing the number of VNs that it can embed in a physical network. Previous work on this topic shall be studied and some of the already existing algorithms enhanced.

Secondly, the necessary mechanisms shall be added to the NVSS so that it can support also cloud resources, more than just network resources. The mapping algorithms to be implemented on the virtualization platform will be selected from previous study and evaluation regarding the integrated mapping process.

On the third place, a feature shall be designed and created so that a user can request a VN without having to design the full network, thus making the process more simple and automatic. This process will be similar to that of creating a VPN: it has as main objective to connect the different sites and resources of the network and to transfer the complexity of network management to the operator.

In order to make all this possible, the main concepts and technologies regarding server and network virtualization will be studied. The current point of situation in terms of existing products of VN and cloud resources will also be explored, and mapping algorithms will be presented, discussed and evaluated. Some experimental results of the NVSS platform will also be presented and discussed in order to evaluate the performance of the new features and feasibility of their deployment in a practical scenario.

1.3 Contribution

As the result of the accomplishment of the proposed objectives, this Master's Thesis produces different contributions.

On the one hand, it presents different methods for mapping virtual networks, with and without cloud resources, together with an evaluation of each of them using a discrete event simulator. The fact that the data of different algorithms is directly comparable allows us to understand better how the mapping of virtual networks should work, and which methods are the best. The proposed and evaluated algorithms show improvements when compared

to the previous work, allowing to enter into consideration with parameters such as link differentiation and the long-range interdependency within the process of node placement.

On the other hand, it enables the NVSS virtualization platform to manage cloud resources. It does this in two ways: by providing the tools for the platform to support and distinguish physical nodes of different types, and by making it map those different resources considering their particular characteristics while still assuring the integration of the network in the mapping process.

Besides this, an alternative method for specifying network requirements was also added to the NVSS. This method only requires the input of data about the endpoints of the network, the virtual resources needed and the amount of traffic going in and out of the network in each end point. With this information the NVSS is now able to automatically design, map and create a VN following this input requirements.

The performance of these features is experimentally tested through experiments that evaluate the quality and speed of the mapping algorithm implemented, as well as the performance of the different methods for VN configuration and creation.

In order to get more data regarding virtualization, an experiment was also made to observe the effect of virtualization on the traffic of virtual networks. This data helps us to understand better the possible consequences of virtualization, its limits and how it can be improved.

A conference paper was already submitted to The Fifth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2011), which introduces the concepts of both network and cloud integration through VNs and VPN approaches, and depicts the first results of the platform. More publications are now being prepared with respect to the mapping algorithms and the traffic analysis.

1.4 Thesis Outline

Chapter 2 will provide the reader with insight on the current virtualization technologies and initiatives, particularly regarding network and server virtualization. It will also present some of the main current commercial offers of virtual networks, connectivity and cloud. The main VN and cloud mapping algorithms in the literature will also be studied, and an overview of the NVSS and other platforms is provided.

In Chapter 3 the current mapping algorithm in use by the NVSS will be evaluated and new proposals to improve the quality of its mapping solutions and to allow it to consider link differentiation and virtual servers mapping will be presented and evaluated through computer simulation testes and results.

By Chapter 4 the advanced mechanisms to be added to the NVSS will be specified, and the interface requirements and use cases shall be considered.

In Chapter 5 the software implementation of the advanced mechanisms designed in Chapter 4 will be explained. This will include a description of the classes and databases, followed by a detailed explanation of the programming structures for each new feature.

Chapter 6 will describe the experimental tests and results obtained on the testbed. The focus will be put on the performance results of the new features, but some results regarding the effects of virtualization on the traffic going through virtual networks will also be gathered and included.

Lastly, Chapter 7 will summarize the work done throughout this thesis, together with

the main knowledge obtained and the summary of the new features included in the NVSS. Some considerations about possible future work areas and paths will also be made.

Chapter 2

State of the Art

2.1 Overview

In this chapter we will introduce the main concepts related with the work developed along this Thesis, as well as do a general survey of the current developments on this Thesis main areas of research. The chapter begins in section 2.2 with a high-level overview of the most important virtualization technologies and initiatives. In the following section, 2.3, some VN mapping algorithms will be presented and discussed, followed by a review of some of the most relevant commercial products on virtual networking and cloud computing in section 2.5. After that, in section 2.6, a set of platforms for the management of cloud are presented. The chapter ends in section 2.7 with different virtualization projects and platforms, with particular emphasis on the presentation of the virtualization platform that was the starting point for the development of the platform presented in this Thesis.

2.2 Virtualization Technologies and Initiatives

2.2.1 Server Virtualization

Server virtualization consists in the full emulation of a server from the point of view of the user, creating a separation layer between the hardware and the server emulation and therefor allowing for the hardware to be shared by a number of virtual servers. It appeared as a solution mainly designed to allow for multiple isolated server systems to coexist in the same physical server, thus allowing for structure cost reduction and improvement in the using experience (which could be simultaneous, instead of time-shared, for instance) [13].

In order to enforce this, a virtualized system includes a new layer of software called the Hypervisor or Virtual Machine Monitor (VMM). Its aim is to achieve dynamic resource sharing and arbitrate access to the underlying physical resources so that these resources can be effectively and efficiently shared among the different Operative Systems (OSs) running over the virtualization layer.

In 1974 Popek and Goldberg defined three main characteristics believed to be essential to the architecture of virtualizable machines [48]:

- *Any program run under the VMM should exhibit an effect identical with that demonstrated if the program had been run on the original machine directly.*

- *A statistically dominant subset of the virtual processor's instructions is executed directly by the real processor.*
- *The VMM is in complete control of system resources.*

The first research Hypervisor (hardware virtualization technique) to offer full virtualization support was implemented on IBM's CP-40 system in January 1967 and it supported multiple instances of client OSs. Virtualization allowed, for example, for beta and experimental OSs to be deployed and debugged without compromising other stable running OSs, which was a great advantage in terms of robustness, stability and infrastructure costs.

Advantages and Disadvantages

Main advantages of server virtualization include: safety, trust and availability, optimize resource utilization, cost, load balancing, legacy applications and versatility.

Some of the main disadvantages of server virtualization are: safety, management, performance and performance.

Due to the enormous advances in computing performance and resource availability since 1967, which made powerful computers not only available to enterprises but also to personal users, virtualization and its advantages have also become more available to personal users.

Server Virtualization Tools

A number of server virtualizations tools are available today, such as OpenVZ, Xen, VMWare Server, Q, Microsoft Virtual Server, Sun Virtual Box or Oracle VM, which are some of the most known.

2.2.2 Network Virtualization

Similarly as for server virtualization, network virtualization consists in the emulation of a computer network, which is experienced by the user in the same way that it would experience a physical computer network. This means that the user does not have access to the underlying computing and network resources and that these resources should be irrelevant and unnoticeable from the point of view of the user of the network. A VN is then built and defined as a group of virtual resources (which can be virtual routers or virtual servers, for example) connected by virtual links.

Network Virtualization (NV) allows an optimized use of the network, consolidating this resource. It also aims at reducing CAPEX and OPEX for operators, and providing them with the possibility to run network protocols independently of the physical network, thus allowing the adjustment of each VN to the services that run on top of it.

Network virtualization techniques should follow some guidelines which can be summarized as follows:

- *Flexibility, Programmability and Heterogeneity.* It should be possible to program every network element and protocol, use any topology and virtualize every type of underlying network technology.
- *Scalability and Manageability.* Multiple VNs should be able to coexist over the same shared physical network, as many VNs as the physical network can accommodate.

The management tasks should be made modular and introduce accountability at every networking layer.

- *Isolation and Security*. The coexisting VNs must be isolated in order not to affect each others performance. Data from one VN should not be controllable by nor visible for any other VN.

- *Legacy Support*. Current network technologies and protocols should be seamlessly upgradeable, and NV must be able to replace the Internet without disrupting it.

Network Virtualization Technologies

There are several network virtualization technologies, such as VPN, MPLS and Virtual Local Area Networks (VLANs).

2.3 Virtual network mapping algorithms

In order to use with maximum efficiency the physical network resources the operator must possess a mapping algorithm that will allow him to decide where is the best location to host virtual nodes and virtual links. The algorithm should be fast and should allow the operator to host the maximum number of virtual networks and virtual resources as possible, as well as to balance the load in both nodes and links.

The greatest concern in these algorithms is due to the need for optimum mapping of both links and nodes simultaneously. This simultaneous optimization is known to be *NP-hard* [12, 65] and therefore is only tractable for a small amount of nodes and links. In order to solve this problem in a load balancing perspective, several approaches have been suggested, mostly considering the version of the problem where the VNet requests are fully known in advance.

Zhu and Ammar et al. [65] propose a heuristic and centralized algorithm for dealing with virtual network embedding. Their approach tries to solve an online version of the problem, considering reconfigurations of the existing VNs, when VN requests arrive. In order to further improve the performance of the basic mapping algorithm, a subdivision technique is also explored. The goal of the mapping algorithm is to maintain a low and balanced stress of both nodes and links of the substrate network; with that goal in mind, the algorithm starts by determining each node's stress (number of virtual nodes running on the substrate node) and the links' stress (number of virtual links whose substrate path passes through each substrate link). With these weights determined, the Neighborhood Resource Availability (NR), that takes into account both the node stress and the local links stress, is calculated for each node. The node with the highest NR is selected as the start node to begin the candidate selection. Next, a set of substrate nodes is determined weighted by their distance to the previously selected substrate node, its node potential is calculated, and in the final step the virtual nodes are mapped. Virtual nodes with more interfaces are assigned substrate nodes with higher NR since virtual nodes with more interfaces are also more likely to setup more virtual links and increase the load on both the substrate node and neighbor links.

Nogueira [41] uses an approach based in Zhu and Ammar et al. [65] but considers the different parameters that characterize a physical and virtual machine such as CPU Load, processor frequency and memory, while still considering the number of virtual machines

running over the physical node. It uses these parameters to calculate the node and link stresses and then uses both node and link stresses to calculate the potential of a node to be chosen as host to virtual node, by multiplying the node stress by the link cost, which is a value composed by the link stresses of the physical paths from the candidate to the virtual neighbor candidates. Since this algorithm is particularly relevant for this Thesis , it can be read in Algorithm 1.

Algorithm 1: Mapping Algorithm Pseudo-Code

input : $Substrate$ (Substrate Network) , $V_{Request}$ (Requested VNet)
output: V_{Map} (Mapped Virtual Network)

foreach *Link i in Substrate.Links* **do**
  $S_{LS}(i) = \sum_{j}^{N_V} \sum_{k}^{Lv_j} ((S_{LV_j}(k_j)|k_j \supseteq i))$;
end
foreach *Node i in Substrate.Nodes* **do**
  $S_{N_i} = \frac{\sum_{j}^{N_V} \sum_{n}^{N_{V_j}} \Lambda(n_j,i)}{\delta + \text{Free RAM·CPU Freq·(N.CPU - Load)}}$;
end
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  foreach *Node i in Substrate.Nodes* **do**
    if *MeetsConstraints(n, i)* **then**
      n.Candidates.Add(i) ;
    end
  end
end
SortVirtualNodes($V_{Request}$) ;
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  foreach *SourceCandidate v in n.Candidates* **do**
    $\pi(v) = 0$;
    foreach *Link k connected to n* **do**
      ConnectedVNode=GetLinkDestination(k) ;
      foreach *DestCandidate u in ConnectedVNode.Candidates* **do**
        D(v,u)= Cost(CSFP_Dijkstra(v,u)) ;
      end
      $\pi(v) = \pi(v) + \sum_{u \in V_C} D(v,u) \cdot S_{N_v}$;
    end
  end
  n.Map $= v : \pi(v) = min(\pi)$;
end
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  V_{Map}*.Nodes* $\cup$ n ;
  foreach *Link k connected to n* **do**
    ConnVNode=GetLinkDestination(k) ;
    V_{Map}*.Links* $\cup$ CSFP_Dijkstra(n.Map,ConnVNode.Map) ;
  end
end

Chowdhury et al. [16] propose different algorithms with better coordination between the node and link mapping phases, by using deterministic rounding techniques in one of them and randomized rounding techniques in the other. The most interesting part of the article is that the evaluation of the algorithms is made in terms of revenue and cost, by using a discrete events simulator to emulate a physical network receiving network requests and then verifying how many resources could be hosted with the different algorithms, and at the same time the VN request acceptance ratio and the ratio of utilization on links and nodes. We believe this is an interesting approach since the evaluation criteria of considering costs and revenues is what will interest the most to the Operator of the network when competing in the market and working to increase profit margins.

M. Melo et al.[39] specifies an integer based optimization formulation to determine the optimal mapping of virtual networks resource, considering resource management aspects and location. This approach is compared with the heuristic in [41]. Lischka [33] proposes a VN mapping algorithm based on subgraph isomorphism detection that maps nodes and links in the same stage.

Lu [34] presents a method for mapping a virtual network where enough capacity is allocated to handle any traffic pattern allowed by a general set of traffic constraints. This paper also attempts to find the best backbone topologies entering into consideration with the substrate network and traffic conditions.

Houidi [29] presents and evaluates a distributed algorithm for load balancing, and node and link mapping. It uses a Multi-agent approach and proposes a VN Mapping Protocol to communicate and exchange messages between agent-based substrate nodes to find an appropriate mapping solution.

Although all these algorithms provide a solution for the virtual network mapping problem, most of them fail to take into consideration that their links may not be constrained only by bandwidth, but also by latency, jitter and loss or other differentiating criteria, such as links requiring high-security guarantees. They also tend to privilege load balancing, with the consequence of sacrificing some extra resources to achieve better load balance.

2.4 Cloud resource management and mapping

Server virtualization has created a path for enabling innovation in cloud products and services, since virtualization allows a physical server to become a container of separately functioning and manageable virtual servers. When virtualizing a server we are, in fact, decoupling functionalities from infrastructure, thus enabling the (virtual) server to be hosted by several different physical machines, with different characteristics and in different places.

Regarding cloud services, from the service user point of view it should be irrelevant where the server the user is accessing is placed. However, from an operator point of view, it is not irrelevant where the virtual server is placed, since this might affect service quality and profits. There are different factors which might influence the choice of where to place one or more virtual servers.

For instance, it is relevant for a certain set of services provided over the web to have low latency, such as real-time multi-player on-line games, among other real-time services [22]. This usually implies that the server should be near the service users, that few connections should be used between the server and the clients, and that these should have small delays in the links and nodes.

On the other hand, placing a virtual server in a large data-center provides a large set of economic benefits, mainly originating from the economy of scale. Large data-centers built and operated at low-cost locations are able to provide a reduction of costs in electricity, network bandwidth, operations, software and hardware [35].

Jurisdiction might also play an important role in the placement of virtual servers. Different countries have different laws regarding internet usage, for example regarding copyrights or freedom of expression. Iceland, for instance, aims at bolstering its economy by making the country an attractive host for Internet content with initiatives such as the Icelandic Modern Media Initiative [30].

Other countries, such as Ireland, have been successful in attracting large web companies

and services in part due to their low taxes. Paying for a virtual server from Amazon hosted in America is cheaper than for one hosted in Europe [11], which shows that there are costs associated with the location of the physical machines, and that these costs should be taken into consideration when placing a virtual server.

We have seen that besides having costs and limitations associated with the location and size of the physical resources, there are also constraints which are dependent on the network. QoS guarantees regarding latency, bandwidth availability or traffic congestion are critical for many services provided through the cloud, thus benefiting from a coordinated decision for virtual server placing that considers both network and servers.

Several papers on Cloud resource management have been written.

[55] presents a study of bin-packing heuristics for resource allocation for Distributed Real-time Embedded (DRE). In [46] a fair joint multiple resource (computational and network) allocation method is presented. The authors propose the enhancement of a method they had previously presented.

[32] addresses the placement decision method taking into account latency and bandwidth constrains in the situation where the user accesses content from several servers.

Energy costs in datacenters represent a major consideration nowadays, due to the very high amounts of energy they use. Reducing them is one of the ways of getting competitive advantage and increasing profit margins, besides creating 'green publicity'.

[20] and [19] address this subject. The first evaluates Power Consumption-Based (PCB) and Transmission Rate-Based (TRB) algorithms for server selection, while the second one proposes an Extended Power Consumption-Based (EPCB) algorithm that proves to be able to reduce the power consumption more than TRB and PCB algorithms.

As for virtual server placement in CC, there has also been some work done. [14] has proposed an optimal allocation approach to choose the best data-center to store the virtual server request by the user, from a pool of multiple data-centers. A dynamic, decentralized and self-organizing approach to the allocation of Virtual Machines (VMs) to physical servers in public and private Clouds is proposed in [17]. The approach is based on a Cross-Entropy Ant System (CEAS), where intelligent agents are used to discover physical servers and make allocation decisions. The system is able to dynamically react to changes in the load of the physical servers, as well as to failures in the physical infrastructure. The mapping of VMs into physical servers is done using near-optimal heuristics.

[31] analyses the interplay between Internet Service Provider (ISP) and content providers. The ISP represents the network part of the problem, while the content providers represent the servers. This paper considers 3 different situations regarding the sharing of information and control between ISP and content providers, concluding that separating server selection and traffic engineering leads to sub-optimal equilibrium, but also that extra visibility might also result in a less efficient outcome.

Although these papers address relevant challenges regarding cloud services, they fail to address several points, particularly regarding the network component of virtual server placing. Most papers either disregard the network or, when considering it, they take into consideration QoS constraints but do not strive to optimize the use of network resources in order to allow for a better embedding of future requests. Also, it is generally considered that virtual server requests are known before-hand, which does not provide a decision method for requests arriving in real-time. Furthermore, the proposed algorithms do not consider the interplay between particular Cloud resources such as CPU capacity, RAM and Hard Disk Drive (HDD) memory.

As we said above, we believe that network consideration and optimization is very important in the virtual server placement decisions. Thus, we believe this is, together with the other aspects that have not been addressed, an opportunity for research, which will be addressed ahead in this document.

2.5 Provision of VPN and cloud resources

In this Thesis, we propose the inclusion of advanced mechanisms for the creation of VNs similar in structure and functionalities to VPNs. Thus, it is important to understand better what VPNs are really about. We also believe it is important to assert the current value and importance of CC for the computing and telecommunication companies and markets.

2.5.1 VPN

Organizations which have premises geographically dispersed frequently need to connect different sites and build a network with strict QoS requirements. Such requirements might include high security guarantees or bandwidth, for instance. Usually, in order to attend to this need, a VPN is deployed, consisting in the emulation of a private network over a physically shared network owned by an ISP. Some of the most popular solutions are Multiprotocol Label Switching (MPLS) VPNs. These VPNs are deployed over Layer 3, as a service provider offers a network-based IP VPN routing and forwarding service across its own IP-based MPLS backbone network [28]. The use of MPLS provides advanced traffic engineering capabilities that, coupled with Internet Protocol (IP) QoS, allow the ISP to offer VPN services complying with demanding QoS requirements. By creating a separate VPN Routing and Forwarding (VRF) instance for each VPN in each Provider Edge (PE) router, the network will be able to perform layer 3 data separation, VPN isolation and private IP address overlapping. This way, a packet entering through a PE router interface associated with a specific VPN will be routed according to the VRF for that VPN. This routing decision will direct the packet to one of the Label-Switched Paths (LSPs) which are set up by MPLS, beginning and terminating at PE routers. Operators generally charge these services according to the Service Level Agreements (SLAs) and charging a surplus for the excess traffic entering the network.

2.5.2 Cloud Computing

Nowadays several enterprises from different countries offer CC. CC is made available as a service, similar to commodities such as water and electricity, with a common a pay-as-you-go payment model. This product is particularly interesting on economical grounds due to the possibility it offers to outsource IT, to reduce management overhead, to extend current limited IT infrastructures, to reduce the entrance barrier (minimum infrastructure entry cost) and to reduce the risk of wasting resources. The possibility to turn CAPEX costs into OPEX costs and the highly scalable provisioning environment are also important features of CC.

It is estimated that the global market for Cloud Computing Technologies represented a total of $21.2 Billion as of 2010. By 2016 it is foreseen that this value will reach $74.9 Billion, with a Compound Annual Growth Rate of 23.4% in the intermediate period, as in Figure 2.1 [36].

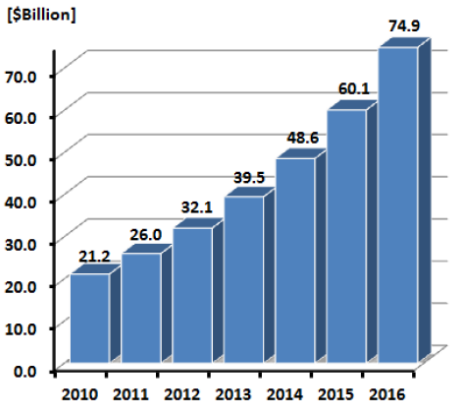


Figure 2.1: Cloud Computing Cumulative Market [36]

Major Market Players and Products

Cloud computing is sold nowadays by a vast range of enterprises over the world. Here we will present some of the most important cloud providers and products: Amazon, Orange and Portugal Telecom.

Amazon

Amazon.com was founded in 1995 as an online book store. Since then it as expanded to sell more types of products [10]. In 2006 Amazon launched Amazon Web Services (AWS), which provides cloud computing power, storage and other web-services on demand on a pay-per-use model [11].

Some of the most relevant services from the AWS suite are [11]:

- *Amazon Elastic Compute Cloud (Amazon EC2)* - A web service that provides resizable compute capacity in the cloud. The user can set the operating system, services, databases and application stack required for the hosted application.
- *Amazon Simple Storage Service (Amazon S3* - A simple web services interface that can be used to store and retrieve large amounts of data, at any time, from anywhere on the web.
- *Amazon Virtual Private Cloud (Amazon VPC)* - A secure and seamless bridge between a company's existing IT infrastructure and the AWS cloud. Amazon VPC enables enterprises to connect their existing infrastructure to a set of isolated AWS compute resources via a Virtual Private Network (VPN) connection, and to extend their existing management capabilities such as security services, firewalls, and intrusion detection systems to include their AWS resources.

Amazon Elastic Compute Cloud (EC2) is classified as a compute product and the client only pays what he or she consumes, like instance-hours or data transfers. Amazon's EC2 price catalog is available in the AWS EC2 web page. Prices vary depending on the characteristics of the machines, on the payment method (on-demand, reserved, spot price), on the

operating system running (Linux is cheaper) and on the region (US is cheaper than Europe, for example). Users also pay for the amount of traffic going in and out.

Amazon Virtual Private Cloud (VPC) is classified as a networking product, as it allows the user to provision a private, isolated section of the AWS Cloud where it can launch AWS resources in a virtual network it defined. Its architecture is depicted in Figure 2.2. The user can define the virtual network topology, select the IP address range, create subnets, and configure routing tables and network gateways. Amazon highlights benefits of the service such as:

- *Multiple Connectivity Options* - The user can connect the Virtual Private Cloud (VPC) to the Internet using public subnets or using Network Address Translation (NAT). It can also connect the VPC to the datacenter using IPsec VPN. Combinations of both methods are also configurable.

- *Secure* - The user can control which machines are accessible and who can access the VPC.

- *Simple* - Amazon provides a Management Console and the user can select one of the common network setups that fits his needs and just press "Create my VPC". Subnets, IP ranges, route tables and security groups are automatically created.

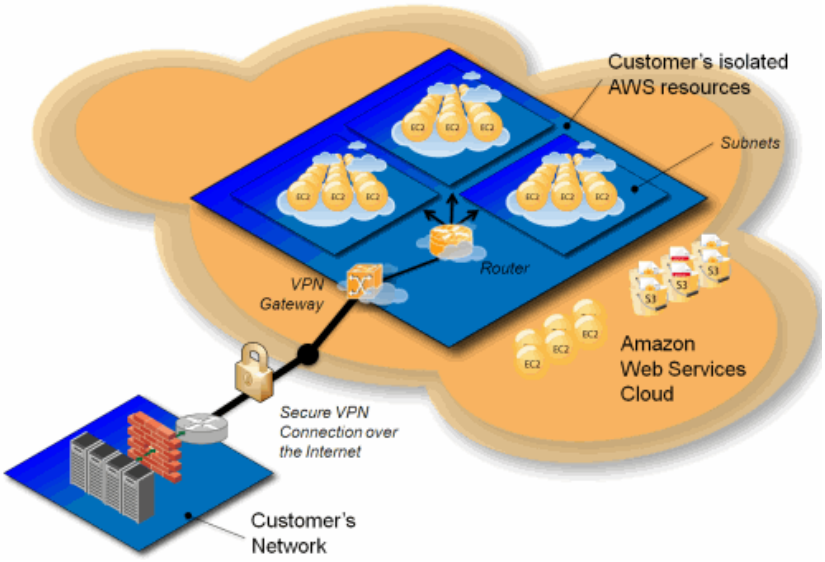


Figure 2.2: Amazon VPC Architecture [9]

Orange

Orange is the main brand of France Telecom [26], which in turn is one of the world's leading telecommunications carriers, with 193 million customers and consolidated operating revenues of 45.5 billion euros for 2010. In June 2009 Orange launched "Flexible Computing", a product to allow businesses to outsource part or all of their IT infrastructures, based

on virtual technology. By December of the same year, it announced it would be offering complete CC services, from infrastructure to real-time business applications [45].

"Flexible Computing" aim is to answer the growing demand for flexibility and adaptability of critical business applications. It addresses the growing demand for performance, availability and security by offering completely customizable virtual server architectures. A screen-shot from this product guided tour is displayed in Figure 2.3.

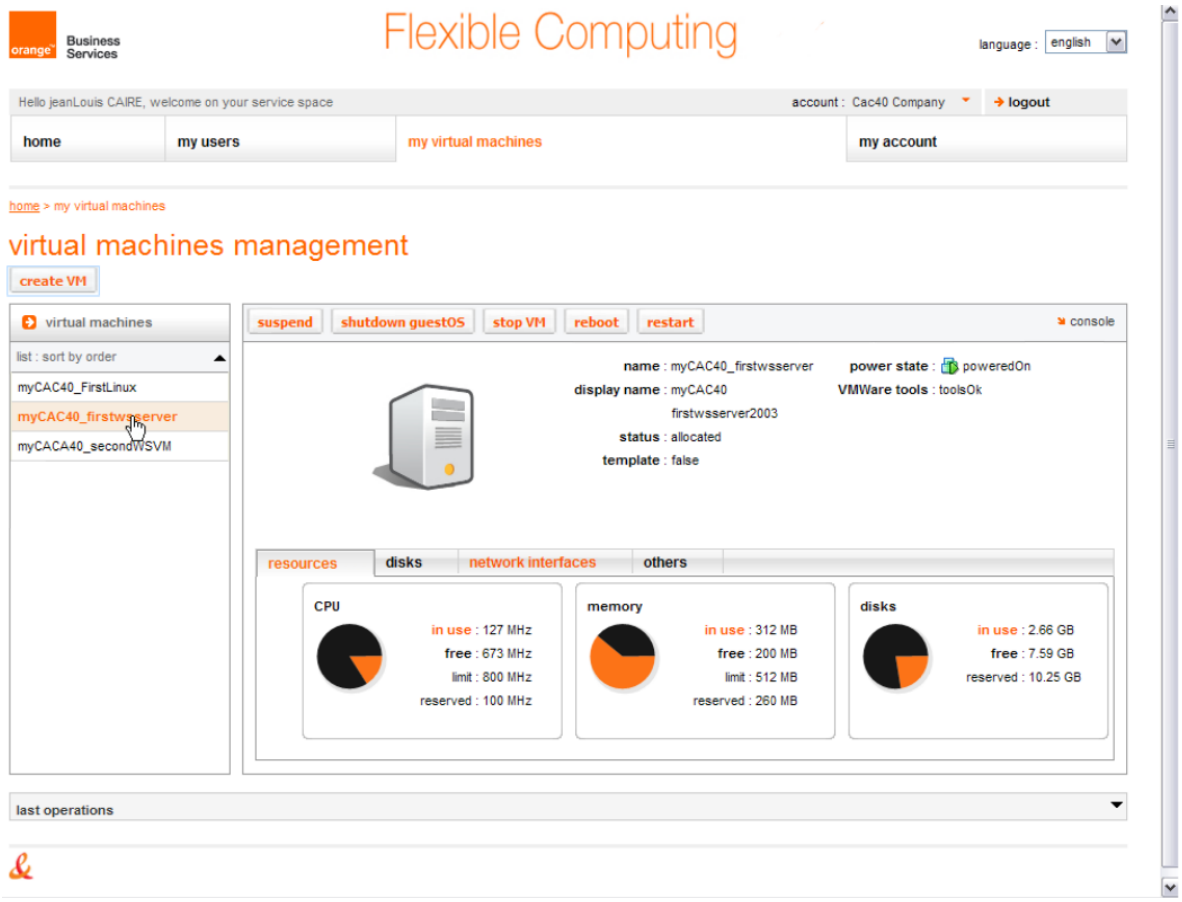


Figure 2.3: Picture from Orange's Flexible Computing guided tour [45]

Orange's IT infrastructure includes 12 data centers worldwide for customer infrastructure and 18 peta octets of storage [45].

Portugal Telecom

Portugal Telecom is the major telecommunications service provider in Portugal. As of 2009 its business volume represented € 6.785 million and the company serviced 72 million clients [49] .

On September 23rd 2010 Portugal Telecom announced that it would be providing cloud services [49]. On May 18th 2011 it launched SmartCloudPT, a set of Cloud computing products directed at companies [49]. SmartCloudPT offers IT infrastructure, development platforms and business applications that can be easily contracted, used an managed, in an on-demand model and, thus, with high flexibility [50].

SmartCloudPT's solution for virtual servers is divided in two [50]: private virtual servers and public virtual servers. Private virtual servers allow the hosting of corporate applications, including: web portals, databases, development environments, corporate management systems, client relation management, and content management. Prices for private virtual servers and service details are not available online.

Modelo de Pagamento Mensal – indexado a Recursos Variáveis

Pacote Base		
Sistema Operativo	Windows	Linux
Processamento	1 vCPU	1 vCPU
Memória	1 GB	1 GB
Espaço em disco	20 GB	20 GB
Cópia de Segurança	1 disponível	1 disponível
Preço	€ 30	€ 29

Adicionais		
Processamento	1 vCPU	€ 6,50
Memória	256 MB	€ 5,50
	512 MB	€ 11,00
	1024 MB	€ 22,00
	2048 MB	€ 44,00
	3072 MB	€ 66,00
Espaço em disco	Blocos de 10 GB	€ 2,50

Figure 2.4: Set of prices for the basic and extra features for public virtual servers in the monthly payment model [50]

Public virtual servers utilization possibilities include: hosting several websites, creating a Domain Name System (DNS) server, executing multimedia applications, creating an e-mail server, sharing movies, games and other resources, creating an File Transfer Protocol (FTP) server and creating test environments for websites and applications. Portugal Telecom offers a range of different settings: OS Windows or Linux, processing until 2 virtual CPUs (vCPUs), RAM until 4 Gigabyte (GB), disc storage until 120GB, unlimited traffic, 1 public IP for internet connection, remote access to the server and a license for 10 domains in Plesk Panel. There are 2 payment models: monthly and annual. The monthly model is a pay-per-use model where there is a price for the whole month but where only used days are charged. In this model, basic characteristics cost € 29/30 per month, and additional processing capacity, memory and disk storage can be added for an extra charge. In the annual model the client pays for a fixed set of resources for the entire year, at a more competitive price that in the monthly model, and with the possibility to coexist with the pay-per-use model [50]. Figure 2.4 shows some of SmartCloudPT price information.

2.6 Cloud Management Platforms

There is a large number of software systems to manage cloud computing resources. In this section we will give a high-level overview on three of them: OpenNebula, OpenStack and EUCALYPTUS.

OpenNebula is an open-source project with the objective of building the industry standard open source cloud computing tool to manage the complexity and heterogeneity of distributed data center infrastructures [44]. OpenNebula's interoperability is defended as one of its main advantages compared to other cloud computing tools. Its design principles are: openness, adaptability, interoperability and portability, stability, scalability and standardization. This platform orchestrates storage, network, virtualization, monitoring, and security technologies to enable dynamic placement of multi-tier services on distributed infrastructures, combining both data center resources and remote cloud resources, according to allocation policies. Figure 2.5 represents OpenNebula functioning model.

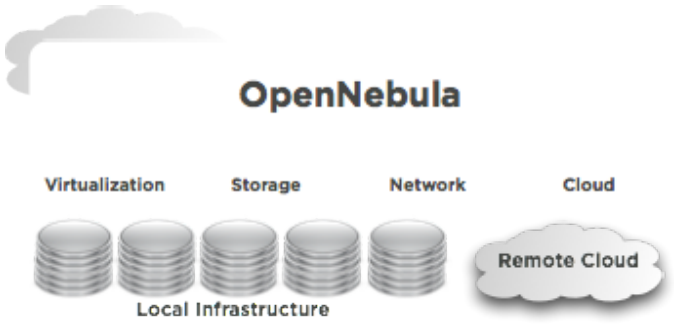


Figure 2.5: OpenNebula's functionality scheme [44]

OpenStack is another open-source project for cloud computing. It aims at building the ubiquitous open source cloud computing platform for public and private clouds. The technology consists of a series of interrelated projects delivering various components for a cloud infrastructure solution. Its main advertised advantages are: control and flexibility, due to its open-source and modular nature; being an industry standard receiving the participation of more than 60 leading companies in the project; being a proven software, since it powers some of the largest public and private cloud in the world; and being compatible and connected [52].

EUCALYPTUS - Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems - is a software infrastructure for implementing cloud computing on clusters. It has two versions: an open-source version and an enterprise version. As differences from other similar software projects: EUCALYPTUS was designed from the ground up to be as easy to install and as non-intrusive as possible, without requiring sites to dedicate resources to it exclusively; the software framework is highly modular, with industry-standard, language-agnostic communication mechanisms; the external interface is based on Amazon's popular interface; it is unique in providing a virtual network overlay that both isolates network traffic of different users and allows two or more clusters to appear to belong to the same Local Area Network (LAN). Overall, this platform's architecture is simple, flexible and modular with a hierarchical design, and the system allows users to start, control, access, and terminate entire virtual machines using an emulation of Amazon EC2's SOAP and "Query"interfaces

[18].

2.7 Network Virtualization Platforms

We have presented technologies to virtualize links and nodes. In this section we will discuss technologies to build virtual networks or, in other words, to create and manage sets of virtual links and nodes that make virtual networks.

There have been several projects regarding network virtualization over recent years, such as: 4WARD[1], AKARI[2], Clean Slate[57], NouVeau[43], Trilogy[61], VNRMS[8], Tempest[4], Netscript[56], Genesis[6], VNET[59], RESERVOIR[54], FEDERICA[24], VIOLIN[7], X-Bone[60], PlanetLab[47], UCLP[62], AGAVE[5], HIPcal[51], GENI[25], GridARS[27], VINI[3] and CABO[21].

In the following subsections we will present the virtualization projects and platforms FEDERICA, GENI, HIPerNET and Network Virtualization System Suite. The latter one, which was part of 4WARD, will be considered in greater detail since it forms the basis for the development of this thesis' platform.

2.7.1 FEDERICA

The FEDERICA (Federated E-Infrastructure Dedicated to European Researchers Innovating in Computing Network Architectures) project has created an European wide "technology agnostic" infrastructure based upon Gigabit Ethernet circuits, transmission equipment and computing nodes capable of virtualization, to host experimental activities on Future Internet architectures and protocols. It is directed at the research community and not industry-driven [24].

Using virtualization, FEDERICA allows research groups to use "slices" of the common infrastructure, which can be configured according to researchers' needs. Among its main objectives, FEDERICA strives to facilitate technical discussions amongst specialists, contribute with real test cases and results to standardization bodies, and integrating the tools and developing a common control plane to manage the infrastructure, as well as the deployment of the necessary protocols to facilitate resource discovery, resource allocation, monitoring, signaling and routing.

2.7.2 GENI

GENI (Global Environment for Network Innovations) is an initiative planned and funded by the National Science Foundation (USA). This initiative includes a large-scale experimental facility to help developing the "future Internet" and a research program strongly coupled to the Future INternet Design (FIND) project. Making use of virtualization, the large-scale experimental facility can be used in a shared way by different research groups, thus facilitating tests of new network technologies which cannot be adequately tested in traditional test facilities.

The GENI architecture is based on 3 layers: User Services, GENI management core (GMC) and Physical Substrate. GMC is the interface between the Physical Substrate (links, routers, processor, storage, etc) and the User Services, which is the layer responsible for slice management and collecting and retrieving measurement data. The GMC defines a stable, long-lived framework while user services and physical substrate can change and evolve [25].

2.7.3 HIPerNET

HIPerNET software is developed within the HIPcal project. This project tackles both network and resource virtualization and the optimization of their use to host VNs. By exploring a different approach to the current services-oriented principles developed in grids to enhance application portability and communications performance control and security, it focuses on developing the appropriate mechanisms to provide a grid substrate based on end to end bandwidth reservation, control overlay, network and system virtualization, among other things [51].

2.7.4 Network Virtualization System Suite

The Network Virtualization System Suite (NVSS) [40] is a platform for virtual network creation, discovery, monitoring and management, which was part of the 4WARD project. It is based in 3 modules: the Control Centre, the Manager and the Agent. This modules are depicted in Figure 2.6.

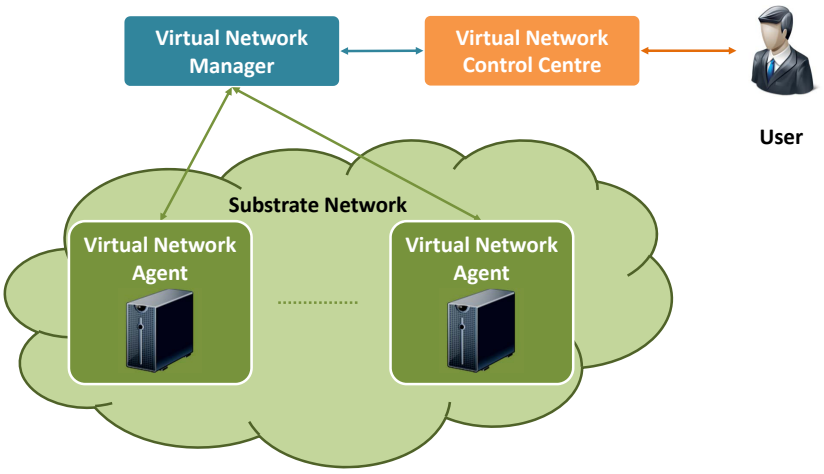


Figure 2.6: Global view of the existing modules.

Control Centre

The Control Centre serves as Graphical Interface for the operator to control the platform. It is programmed in Java and allows the user to design, configure, create, modify, manage and monitor virtual networks through a graphic environment similar to that of a network simulator.

Manager

The Manager is the central module of the NVSS. It is his function to collect information of the virtual and physical networks from the Agents and to send them to the Control Centers. It also receives orders from the Control Centers, processes the requests if necessary and sends the adequate control orders to the Agents.

Agent

The Agent is a module that is present in each physical node and whose function is to gather data about the nodes physical and virtual resources which it then sends to the Manager and to enforce the orders from the Manager, such as virtual nodes or bridge creation, for example.

Main Features

There are four main features in the NVSS:

- *Physical and Virtual Topology Discovery* - In order to gather the information about the topology of the physical network and of the virtual networks running over it, a distribute algorithm is used. Each node, through the corresponding Agent, gathers data about itself and about the neighbor nodes. It then sends it to the Manager which can easily integrate the information from the different Agents and have the full virtual and physical networks topology and other information available. A trigger mechanism is used so that when an event occurs in the configuration of the resources or in their state, the reaction is immediate, thus allowing a fast detection of the changes in the topologies.

- *Virtual Network Creation* - Through the Control Center it is possible to design, configure and create virtual networks, using a simple drag-and-drop process of the virtual elements and then connecting them through virtual links. During this process it is possible to configure several node parameters such as CPU, RAM, location, name and description. As for the links, parameters such as latency, jitter and bandwidth are also configurable. When the network is fully designed it can be committed to the Manager, which will use the data gathered from the Agents to decide which physical nodes can host the virtual nodes and which physical nodes and links are the best to map the virtual nodes and links. This process follows the Algorithm 1 described in section 2.3.

- *Virtual Network Monitoring* - The platform periodically verifies the state and configuration of the physical and virtual resources, using the Agents. If any change is detected, the Agents communicate it to the Manager which will send it to the Control Centre so that it is possible to visualize these changes. It is also possible to monitor the state of the nodes, interfaces, links and also CPU load almost in real time.

- *Virtual Network Management* - It is possible to change the state of the virtual resources. A virtual node might be turned off, restarted, suspended or destroyed, using a context menu provided for each resource. Real time change of the RAM allocated to a specific virtual node is also enabled.

The remaining main characteristics can be summarized as follows [42]:

- *User Classes* - This platform is designed to be used by network administrators with total security clearance and access to the substrate network. The user should be experienced in Linux environments and networking.

- *Operating Environment* - This platform is designed to run on Fedora Core 8 and Debian Lenny Linux distributions with the Xen kernel.

- *Constraints* - The C programming language was used for programming every module except the GUI, which was programmed in Java.
- *Dependencies* - In order to be able to properly run the software, except the GUI, the following modules must be installed or enabled: libvirt, libxml2, glibtop2, bridge-utils, 802.1q module. To properly run the GUI the Java runtime environment should be installed.
- *Performance Requirements* - The designed platform has a low overhead, and consequently, a small performance impact on the substrate nodes and network. The CPU and RAM usage is kept to a minimum.
- *Security Requirement* - Due to its deployment features, the platform, or a part of it, must be run with elevated privileges, i.e. in root mode, and is therefore potentially dangerous for the system if it becomes compromised or misbehaves.
- *Software Quality Attributes* - This software was designed in an extensible and modular manner; new features may be added without significant changes to the underlying architecture and isolated; independent tests on some of the features may be performed.

2.8 Conclusions

After making this brief listing of the major current developments in virtualization of network and servers, as well as CC products and research, one can see that even though CC is already on the market, it lacks guarantees regarding the network performance, relying mostly on the Internet service, which usually operated in a best-effort mode. This way, integration of both virtual networks and virtual cloud resources in a single integrated service can be seen as an opportunity to provide those network performance guarantees while at the same time allowing for a double optimization of the use of network and resources.

In the following chapters we will explore how network and cloud mapping might be done in an integrated and optimized way, as well as present a virtualization platform capable of managing together network resources and cloud resources.

Chapter 3

Mapping Algorithms

3.1 Introduction

In section 2.3 we presented the state of the art in VN and cloud resources mapping. In this chapter we try to improve those mapping algorithms by proposing and testing by simulation new additions and changes to those algorithms, specifically to the one presented by Nogueira in [41].

In section 3.2 we argue that an effective algorithm for node placement should take into consideration the combined placement possibilities and limitations along all the network and not just among neighbor virtual nodes. In order to do this, we propose an improved algorithm that removes candidates that represent inadequate choices and a back-tracking mechanism to correct wrong choices.

In section 3.3 we propose new formulas to calculate node and link stress in order to improve the final mapping results. In 3.4 we propose formulas to calculate the stress of physical servers, so that the algorithm presented in [41] can be improved in order to consider the mapping of cloud resources, more specifically virtual servers, together with VN routing nodes and links.

3.1.1 Discrete event simulator

To evaluate the results of the proposed changes and additions mentioned in section 3.1 a discrete event simulator inspired by Chowdhury et al. [16] was built, which allows an evaluation of mapping algorithms considering parameters that are related with costs and revenues through a simulation of a scenario where requests for VNs come and go, similarly to what would happen in a real environment. The goal of this simulator is to recreate the environment of an operator that possesses a physical network and receives VN requests to be mapped. The physical resources are gradually occupied over time, with the allocation of RAM and link bandwidth, among other things, to virtual resources and networks. Mapping decisions are made and VNs might be or not accepted depending on the amount of free resources available and the mapping algorithms finding a viable mapping solution or not. The averaged time use of the physical resources is registered so that final statistics about the use of the physical resources can be produced.

This simulator randomly generates a physical substrate, according to a set of input parameters, and it also generates a set of VNs to be mapped, with a corresponding set of

arrival times and lifetimes of the VNs (with the time intervals between arrivals and VN lifetimes being random variables exponentially distributed). In each repetition different physical substrates and VN sets are generated, but the same substrates and VN sets are used for the different mapping algorithms that are being compared. This simulator provides as results the averaged time values for parameters such as the acceptance ratio of VN requests, RAM in use, HDD memory in use, Load in use, virtual machines per node, occupied bandwidth, mapped virtual bandwidth and the mapping time for the algorithms. This process is summarized in Figure 3.1.

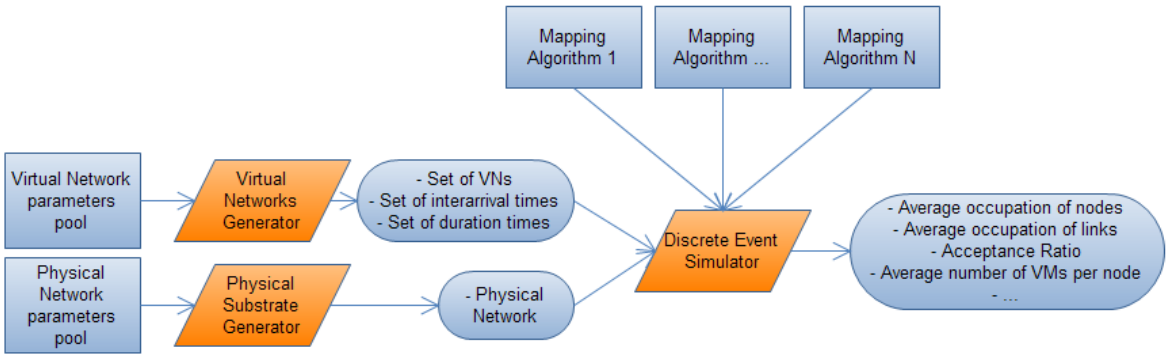


Figure 3.1: Flow Diagram for the Discrete Events Simulator.

In each of the following sections we included some results from the tests made using this simulator in order to compare different algorithms and cost calculation formulas. For each section, a central case scenario was designed and different parameters (e.g. frequency of VN requests, number of nodes of the physical network) were separately changed in order to better evaluate the performance of the different algorithms.

3.2 VN Mapping - Considering node placement interdependency

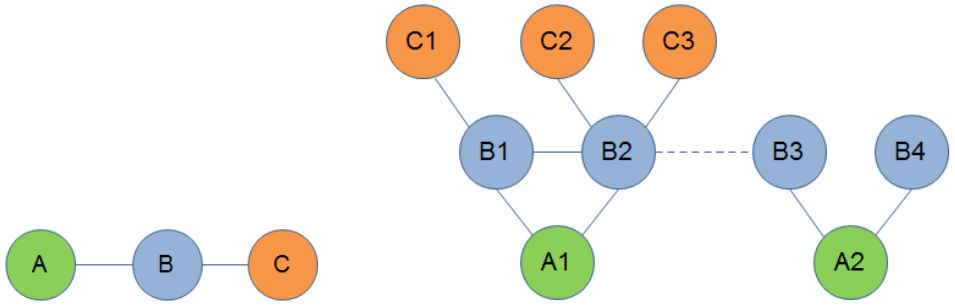
The algorithm proposed by Nogueira [41] and presented in the State of the Art chapter tries to optimize node and link placement simultaneously, considering a cost value for the placement of each node in a certain candidate host and link costs to connect to the virtual neighbors candidates. This kind of method takes into consideration, for each virtual node, the placement of this node and its virtual neighbors, but fails to consider the complexity that large networks present, where the placement of a virtual node might critically effect on the placement of another virtual node that is several virtual links away. For instance, if a physical network includes several links that are not able to accommodate certain or all virtual links (e.g. virtual links with large bandwidth requirements, links with more restrictive security requirements, etc), these algorithms might fail to find a mapping solution.

Our proposal is to start from the mapping algorithm presented by Nogueira [41] and make it aware of this dependence. This way, we aim to remove from the candidate list of each virtual node the candidates that cannot be used as hosts for that virtual node. We also propose a back-tracking mechanism to correct wrong placement choices.

3.2.1 Interdependency mapping

The algorithm proposed by Nogueira [41] starts by producing a list of physical nodes that possess the adequate hardware features to host each virtual node: the virtual node's candidate list. But, since we are dealing with networks, one should also consider that for a physical node to be a candidate to host a certain virtual node it needs to be able to establish a physical connection with at least one of the candidates of each of the virtual node's neighbors, according to the QoS requirements of the virtual link between them. Otherwise it should not be considered a candidate, and it should be removed from the virtual node list of candidates. This removal also implies that other candidates that could only connect to this candidate in order to have a link to the virtual node for which the removed candidate was a applying should be removed as well, and so on.

We propose that, for each virtual link, the algorithm verifies if there is at least one physical path available, with the virtual link QoS requirements, between each pair of candidates to the virtual source and destination of the link. These possibilities of connection are registered and the candidates that don't possess at least one possible connection to one candidate of each virtual neighbor node to the virtual node they are applying are removed. Each candidate removal along the mapping algorithm is followed by a check of the remaining possible connections for the candidates that had possible connections to the removed candidate, and a removal of those candidates will take place if appropriate, and so on. Let's take the example of a VN and a physical network in which we want to map it, see Figure 3.2 :



(a) VN - example (b) Physical Network - example

Figure 3.2: Example of a VN and the Physical Network in which is being mapped

Here we consider that we have a VN made up of 3 nodes - A, B and C - each of them having a set of candidates which is represented with their letter and numbers in Figure 3.2(b). We consider that the physical link between B2 and B3 is unfit to host any virtual link (the reason is irrelevant). If we would map the possible connections between each pair of candidates to each pair of virtual neighbor nodes we would get a representation as depicted in Figure 3.3.

In this case, every candidate node would be checked up after the mapping of the possible connections. Those that didn't have at least one connection to other virtual neighbor nodes candidates would be removed. This is the case of candidates B3 and B4 who don't have any possible connection to any candidate of C, which is a virtual neighbor of B. So, B3 and B4 would be removed. As a consequence, A2 would also not have any connection to

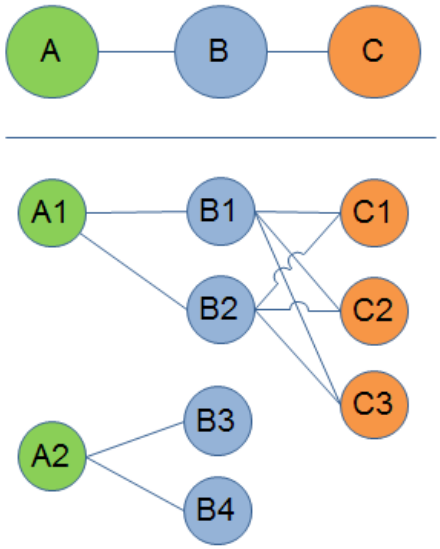


Figure 3.3: Representation of the possible connections

any candidate of node B, the neighbor of virtual node A, thus eliminating candidate A2 too. Since now we only have candidate A1 for virtual node A, A will be mapped to A1. The following virtual node with less candidates is B. We can choose B1 or B2 to host B. If we choose B1, B2 will be removed together with its connections to A1 and to C1, C2 and C3. But these candidates won't be removed since they still have possible connections to the virtual neighbors candidates. Afterwards, a physical host for node C is chosen, for example, C2. If we had not used this procedure to remove unfit candidates, we might have started by selecting A2 to host the virtual node A. Afterwards we would choose C1, C2 or C3 to host C. But when we got to B we would see that none of the candidates to B would be able to connect to A2 and to C1, C2 or C3 simultaneously, thus we wouldn't find a mapping solution.

3.2.2 Back-tracking mechanism

Although this method removes several inadequate candidates, there are some situations where inadequate candidates are not found to be inadequate before they are chosen as hosts. What happens is that, due to the complexity of the network of possible connections, these nodes can only remain as possible candidates to a certain virtual node if other candidates to the same virtual node remain as candidates. Therefore, if one of these nodes is chosen as host for the virtual node, other candidates to the virtual node are promptly removed and the subsequent process of removals will make the chosen host to be unable to connect to other candidates, since those candidates that it could connect to were also removed. This will render the candidate unable to host the virtual node and empty the list of candidates of that virtual node, making all other candidates to all other virtual nodes inadequate candidates.

For example, if we have a case where the representation of possible connections is as in Figure 3.4, none of the candidates will be removed at the start check. But, if we choose candidate A1 to host A, A2 and A3 will be removed and, as a consequence, all other candidates to all other nodes will be removed too, thus making A1 an unfit choice.

We propose that, before selecting a candidate node as host for a certain virtual node, the

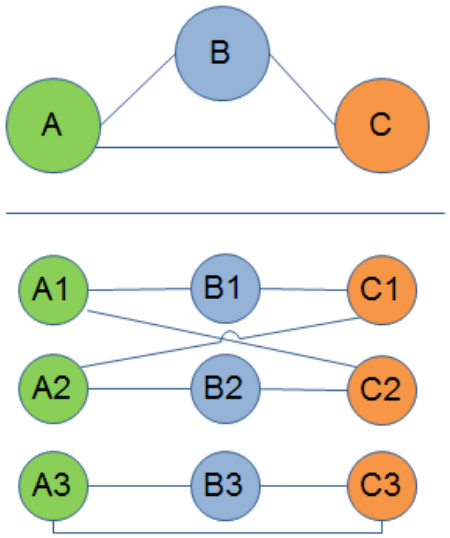


Figure 3.4: Example of a matrix of possible connections for a certain VN and a physical network

candidates list of each virtual node and data of the current possible connections are saved. If the selection of this candidate makes all other candidates for all other virtual nodes inadequate, the data is restored and the chosen candidate, now known to be inadequate, is removed from the candidate list and a new candidate is chosen. If there are no more candidates the algorithm ends without finding a mapping solution.

Using this method for the case of Figure 3.4, it would mean that after selecting A1 the data of the possible connections would be saved. Upon seeing that A1 was an unfit choice, the data of the possible connections would be reloaded, and A1 would be removed, thus making A3, B3 and C3 the mapping solution for A, B, C, which is in fact the only mapping solution available. This process is represented in Figure 3.5. The complete pseudo-code of the proposed algorithm can be seen in Algorithm 2.

In order to verify the impact of these changes in the algorithm, we used the discrete event simulator presented in 3.1.1, while changing variables such as the number of virtual and physical links requiring or offering high security, the rate of VNs to map per time unit and the number of nodes in the physical network. The mapping algorithm by Nogueira [41] is referred as "Short Interdependency", since it only considers the virtual nodes neighbors when mapping a virtual node. The Algorithm proposed here and described in Algorithm 2 is referred as "Long Interdependency" since it considers the influence of all virtual nodes placement on each other's placement.

The parameters used to create the physical substrates and the VNs for the central scenario are summarized on tables 3.1 and 3.2. In this section the parameters for the frequency of VN requests, number of substrate nodes, and amount of links with high-security characteristics were changed one at a time while all others remained the same as they were in the tables describing the Simulation Scenario 1 values. Confidence intervals are for 90% probability. We consider that physical link with high security are reserved for virtual links demanding high security levels, and that a virtual link with high security can only be mapped over a physical path made of links offering high security levels. Average VN life time is of

Algorithm 2: Pseudo-Code of the Mapping Algorithm considering Long Interdependency

input : $Substrate$ (Substrate Network) , $V_{Request}$ (Requested VNet)
output: V_{Map} (Mapped VN)

foreach *Link i in* $Substrate.Links$ **do**
  $S_{LS}(i) = \sum_j^{N_V} \sum_k^{Lv_j} ((S_{LV_j}(k_j)|k_j \supseteq i))$;
end
foreach *Node i in* $Substrate.Nodes$ **do**
  $S_{N_i} = \frac{\sum_j^{N_V} \sum_n^{N_{V_j}} \Lambda(n_j,i)}{\delta + \text{Free RAM}\cdot\text{CPU Freq}\cdot(\text{N.CPU - Load})}$;
end
foreach *Node n in* $V_{Request}.Nodes$ **do**
  foreach *Node i in* $Substrate.Nodes$ **do**
    if *MeetsNodeConstraints(n, i)* **then**
      n.Candidates.Add(i) ;
    end
  end
end
foreach *Link v in* $V_{Request}$ **do**
  foreach *SourceCandidate s in* $Link(v).SourceNode.Candidates$ **do**
    foreach *DestCandidate d in* $Link(v).DestNode.Candidates$ **do**
      if *MeetsLinkConstraints(v,Path(CSFP_Dijkstra(s,d)))* **then**
        PossibleConnections(Link(v).SourceNode, s, Link(v).DestNode, d) = true;
      else
        PossibleConnections(Link(v).SourceNode, s, Link(v).DestNode, d) = false;
      end
    end
  end
end
foreach *Node i in* $V_{Request}$ **do**
  foreach *Node j connected to Node i* **do**
    foreach *SourceCandidate s in* $Node(i).Candidates$ **do**
      if *NumferOf(PossibleConnections(i, s, j, j.candidates)==true)==0* **then**
        RemoveCandidate(i, s);
      end
    end
  end
end
while $\exists$ *Node x in* $V_{Request}.Nodes \mid$ *NumberOf(Node(x).Candidates) >1* **do**
  n = SelectUnmappedNodeWithLessCandidates($V_{Request}.Nodes$);
  foreach *SourceCandidate v in n.Candidates* **do**
    $\pi(v) = 0$;
    foreach *Link k connected to n* **do**
      ConnectedVNode=GetLinkDestination(k) ;
      foreach *DestCandidate u in ConnectedVNode.Candidates* **do**
        D(v,u)= Cost(CSFP_Dijkstra(v,u)) ;
      end
      $\pi(v) = \pi(v) + \sum_{u \in V_C} D(v,u) \cdot S_{N_v}$;
    end
  end
  n.Map = $v : \pi(v) = min(\pi)$;
  SaveData(ListsOfCandidates, PossibleConnections);
  foreach *Candidate j in n.Candidates* $\setminus$ *n.Map* **do**
    RemoveCandidate(i,j);
  end
  if *NumberOf(i.Candidates)==0,* $\forall$ *i in* $V_{Request}.Nodes$ **then**
    RestoreData(ListsOfCandidates, PossibleConnections);
    RemoveCandidate(i,i.Map);
  end
end
foreach *Node n in* $V_{Request}.Nodes$ **do**
  $V_{Map}.Nodes \cup n$;
  foreach *Link k connected to n* **do**
    ConnVNode=GetLinkDestination(k) ;
    $V_{Map}.Links \cup$ CSFP_Dijkstra(n.Map,ConnVNode.Map) ;
  end
end

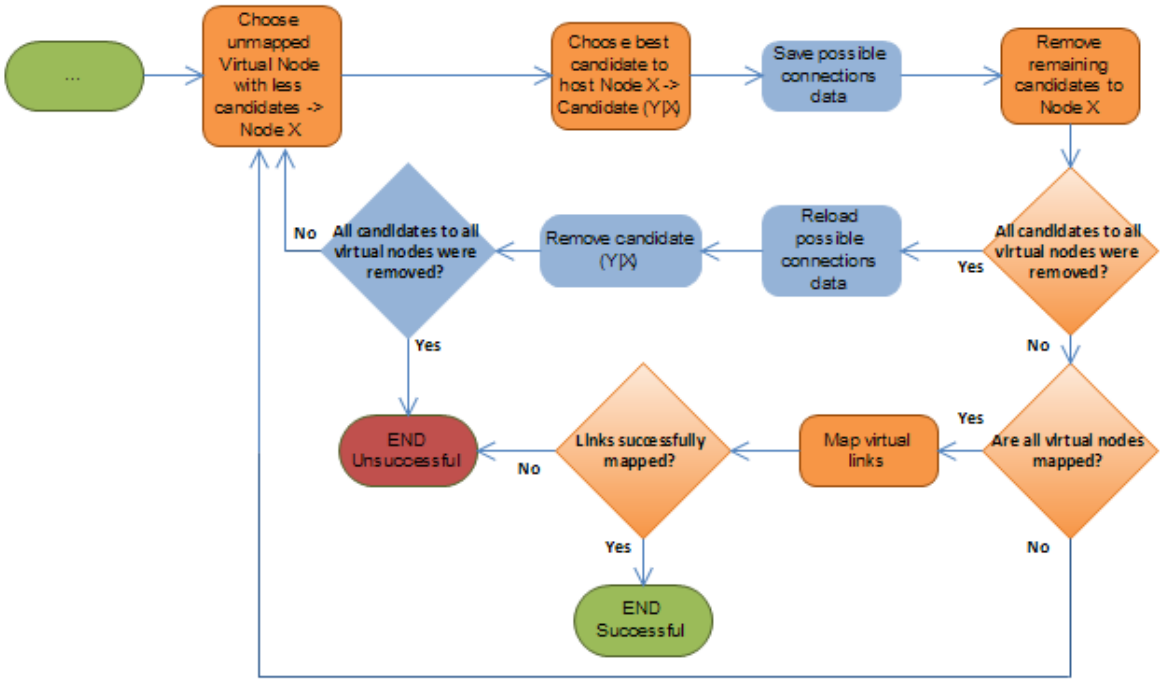


Figure 3.5: Flow Diagram of the last part of the proposed mapping algorithm including the backtracking mechanism (in blue).

---

20 time units. Unless explicitly changed, 2 VNs are requested per time unit on average, and the physical network is made of 35 nodes.

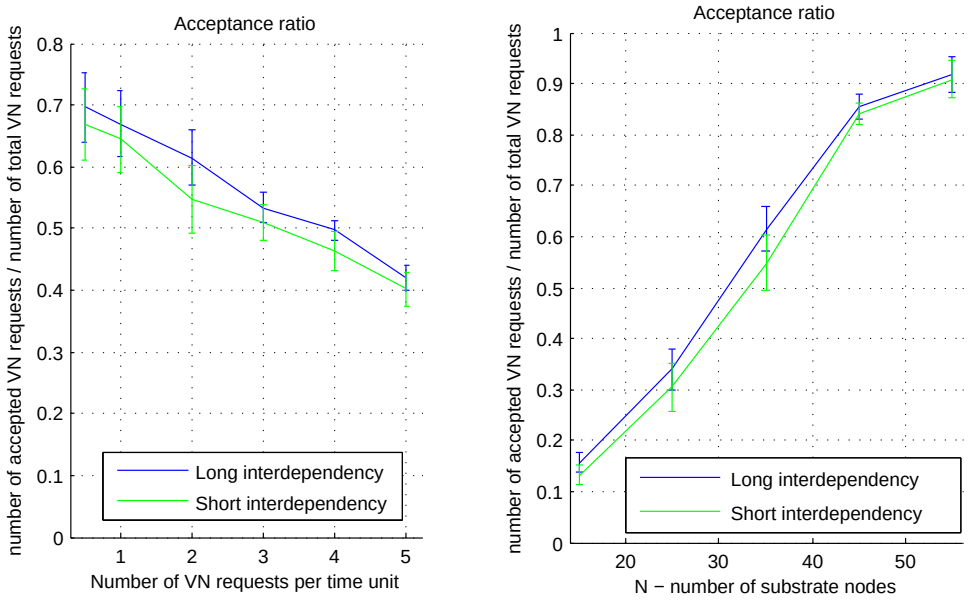
N. CPUs	{2; 4; 6; 8}
CPU Frequency (GHz)	{2.0 to 3.2 in 0.2 steps }
RAM Memory (GB)	{2; 4; 6}
Link Bandwidth (Mbps)	{800; 1200}
High security links (%)	{30}

Table 3.1: VN Mapping Simulation Scenario 1 - Physical Network Nodes and Links' parameters pool.

N. CPUs	{1; 2; 3; 4 }
CPU Frequency (GHz)	{2.0 to 3.2 in 0.1 steps }
RAM Memory (MB)	{64; 128; 256; 512 }
Link Bandwidth (Mbps)	{34.368 139.264 }
High security links (%)	{30}

Table 3.2: VN Mapping Simulation Scenario 1 - VN Nodes and Links' parameters pool.

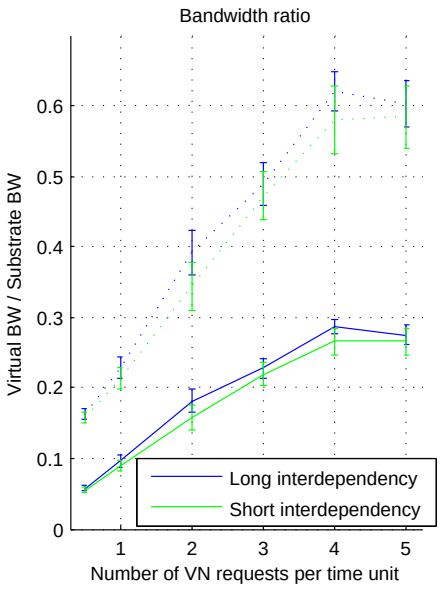
3.2.3 Simulation Tests and Results



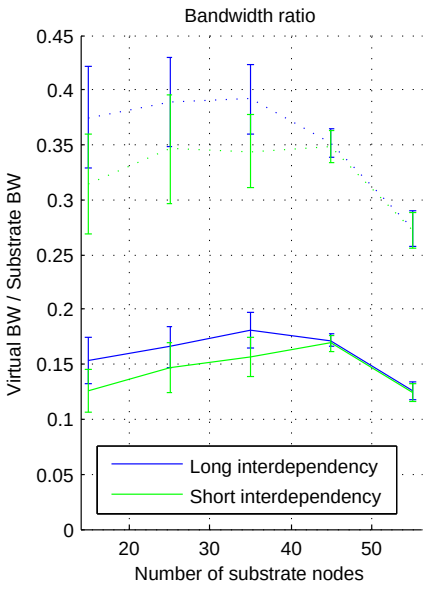
(a) Acceptance Ratio vs. VN Request Rate

(b) Acceptance Ratio vs. Number of Substrate Nodes

Figure 3.6: Rate of VNs requests accepted for the algorithms with Short Interdependency and Long Interdependency

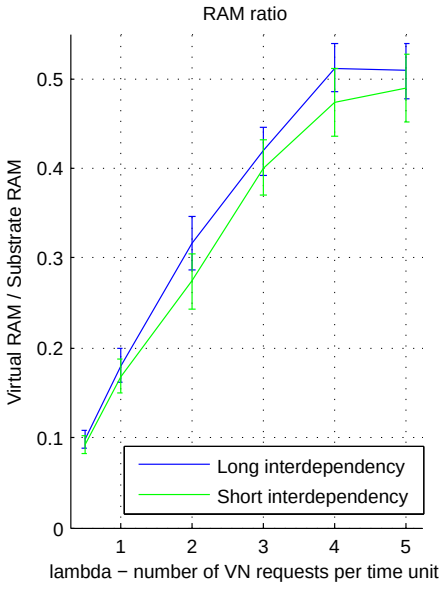


(a) Bandwidth Ratio vs. VN Request Rate

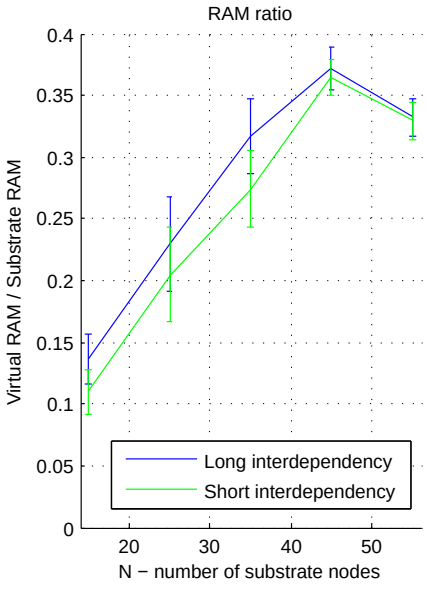


(b) Bandwidth Ratio vs. Number of Substrate Nodes

Figure 3.7: Ratio of Virtual BW in use over Substrate BW for the algorithms with Short Interdependency and Long Interdependency

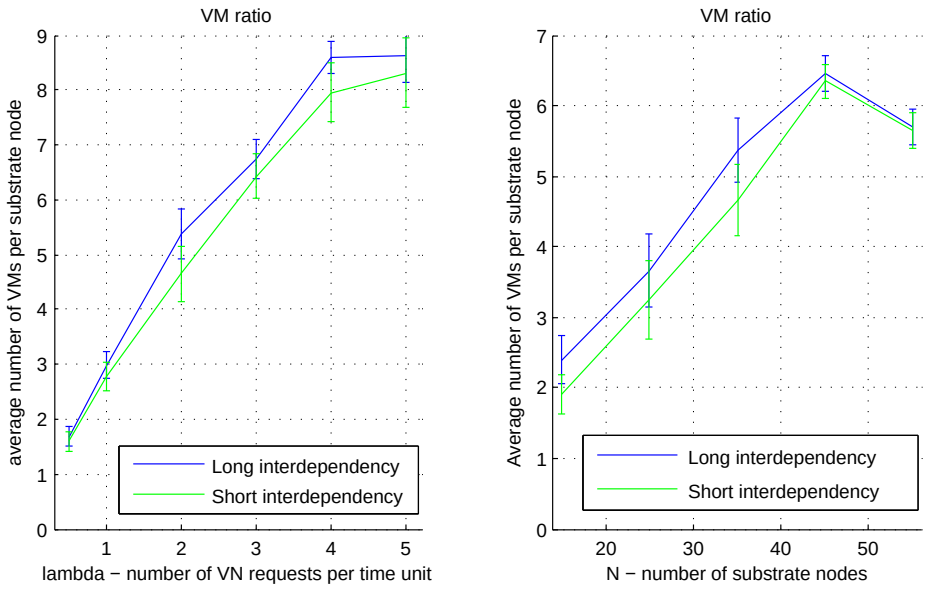


(a) RAM Ratio vs. VN Request Rate



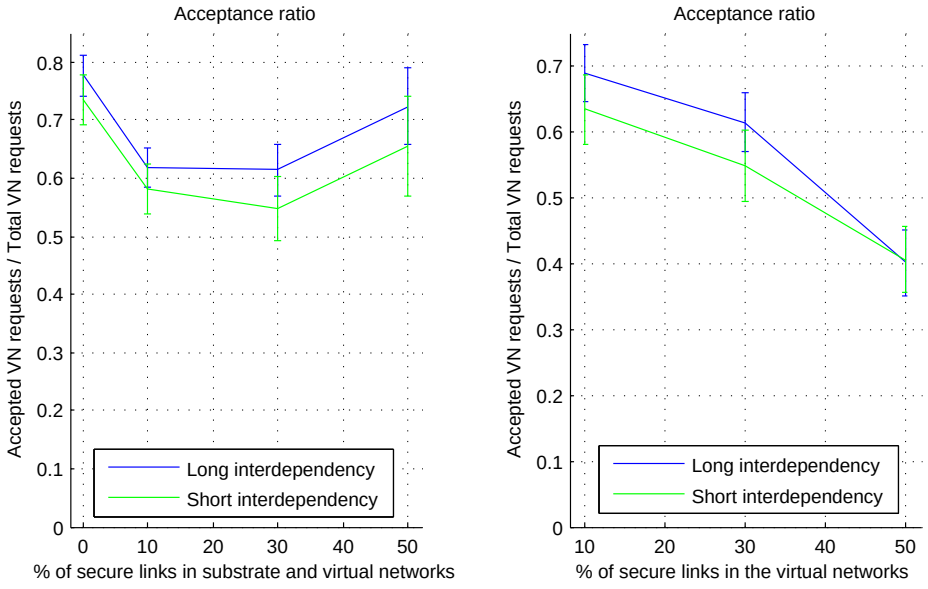
(b) RAM Ratio vs. Number of Substrate Nodes

Figure 3.8: Ratio of RAM in use over Total Substrate RAM for the algorithms with Short Interdependency and Long Interdependency



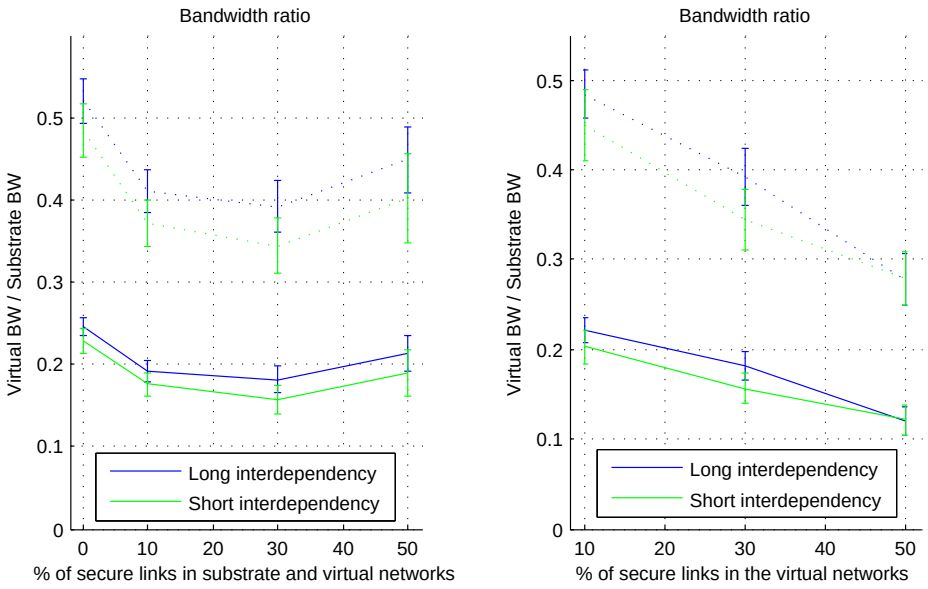
(a) VM Ratio vs. VN Request Rate (b) VM Ratio vs. Number of Substrate Nodes

Figure 3.9: Average Number of VMs per Substrate Node for the algorithms with Short Interdependency and Long Interdependency



(a) Acceptance Ratio vs. % of Secure Links (b) Acceptance Ratio vs. % of Secure Virtual Links

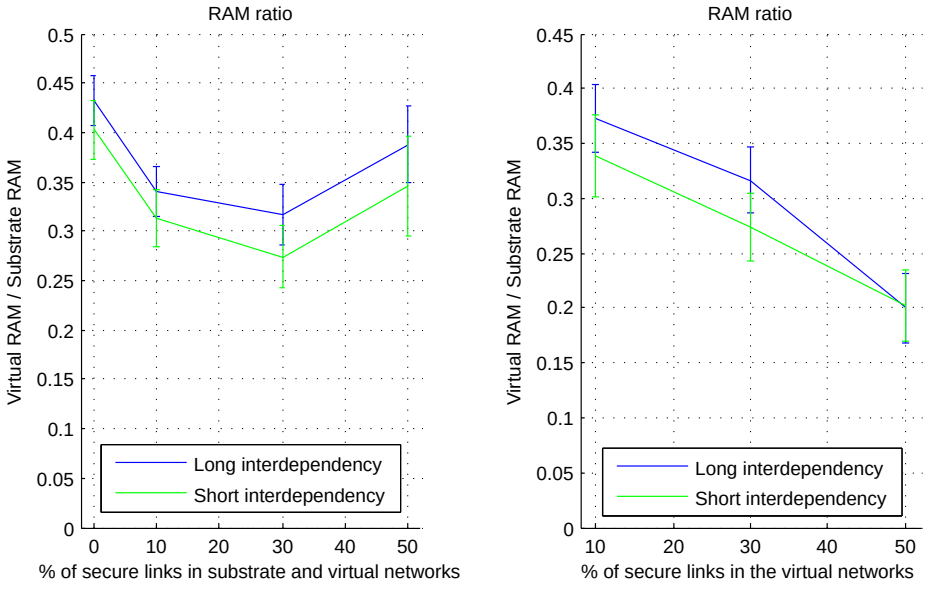
Figure 3.10: Rate of VNs requests accepted for the algorithms with Short Interdependency and Long Interdependency



(a) Bandwidth Ratio vs. % of Secure Links

(b) Bandwidth Ratio vs. % of Secure Virtual Links

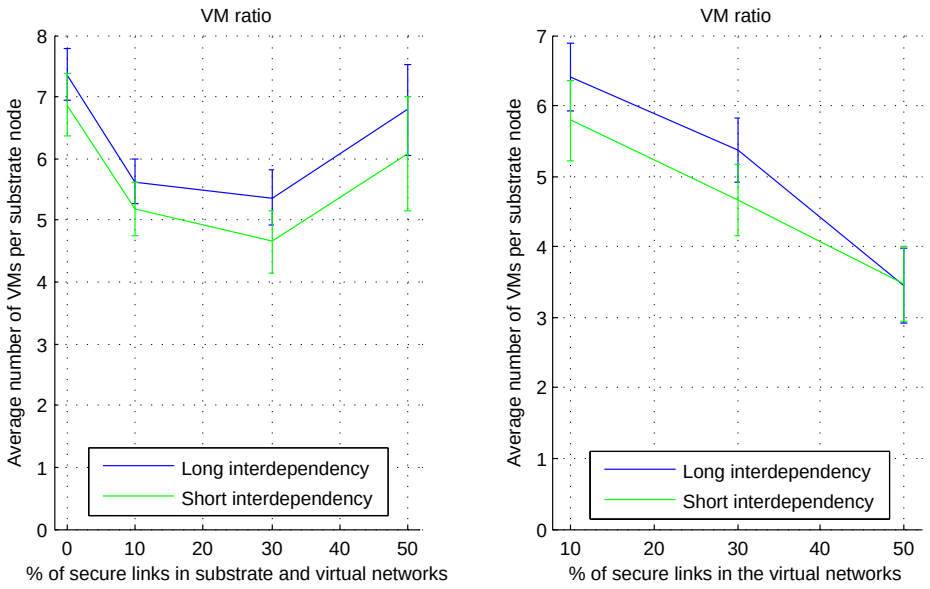
Figure 3.11: Ratio of Virtual BW in use over Substrate BW for the algorithms with Short Interdependency and Long Interdependency



(a) RAM Ratio vs. % of Secure Links

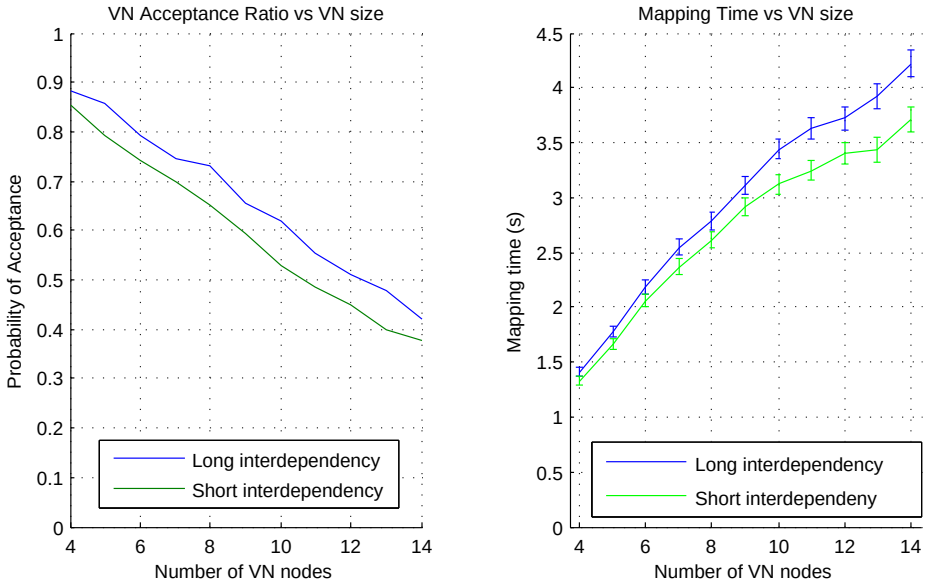
(b) RAM Ratio vs. % of Secure Virtual Links

Figure 3.12: Ratio of RAM in use over Total Substrate RAM for the algorithms with Short Interdependency and Long Interdependency



(a) VM Ratio vs. % of Secure Links (b) VM Ratio vs. % of Secure Virtual Links

Figure 3.13: Average Number of VMs per Substrate Node for the algorithms with Short Interdependency and Long Interdependency



(a) Acceptance Probability vs. Number of VN nodes (b) Mapping time vs. Number of VN nodes

Figure 3.14: Acceptance probability and mapping time vs VN size for the algorithms with Short Interdependency and Long Interdependency

3.2.4 Discussion

In figure 3.6 we have plotted the results for the acceptance ratio, which is the number of VN requests accepted, this meaning that they were successfully mapped and embedded, over the total number of VN requests. First of all we can see that the acceptance ratio decreases as the frequency of the VN requests increases. This is due to the increase in the number of VN requests, which is larger than the increase in number of accepted requests. In spite of this the real number of accepted requests increases. One can also see in Figure 3.6(b) that the acceptance ratio increases as the number of substrate node increases. This is very expectable, since the increase in the amount of node resources and link resources will allow for more VNs to be embedded in the physical network. If one compares both algorithms, it is possible to see that the algorithm that considers the long interdependency of the node placement is able to map the VNs in a way that allows the physical network to embed more VNs than the algorithm which only considers the interdependency of the node neighbors placement.

Although the ratio of accepted requests is an important parameter it should not be considered alone, since it does not distinguish the mapping and embedding of small VNs from the large ones. This is relevant since the amount of virtual resources in use will also correspond revenues and costs for the infrastructure operator.

The first additional parameter we consider is the bandwidth ratio. The bandwidth ratio is the ratio of the virtual bandwidth of the embedded VNs and the maximum amount of bandwidth that the links of the physical network can provide. It is calculated as the sum of the bandwidth of the virtual links for the VNs, and as the sum of the bandwidth of the physical links for the physical networks. In Figure 3.7 we can see in continuous lines the amount of virtual bandwidth of the embedded VNs divided by the total bandwidth of the substrate network. In dotted lines we can see the amount of bandwidth of the substrate that is being used by the VNs, divided by the total bandwidth of the physical network. This values are not the same because a virtual link can span through one or more physical links. It also means that the bandwidth in use on the substrate will always be larger than the bandwidth of the virtual links it hosts. This provides a great opportunity for optimization, since a better mapping should be able to reduce the amount of substrate bandwidth necessary to host the virtual links. The same is not possible to do for the node resources since, for example, 3 GB of RAM for a virtual resource will always require 3 GB of RAM in the physical network.

In Figure 3.7 we can see that the bandwidth ratio, just as the occupied bandwidth of the substrate, increases together with the frequency of the VN requests. As the number of accepted requests increases, there are also more VNs embedded and more virtual bandwidth is being hosted. A different behavior occurs when increasing the number of substrate nodes. First, the bandwidth ratio increases, then it stabilizes and then it decreases. It is not hard to understand why this happens. When the number of physical nodes is small a lot of networks can't be mapped and are rejected, especially the larger ones that represent greater amounts of virtual bandwidth. When the number of physical nodes increases, the number of physical links and substrate bandwidth also increases, but the amount of VNs and virtual bandwidth that becomes embeddable increases even more, therefore increasing the bandwidth ratio. As the number of substrate nodes continues to grow, most VNs are already being mapped, so the increase in substrate bandwidth will not cause a proportional increasing of the virtual bandwidth being mapped, which causes the bandwidth ratio to fall.

Moreover, we can also observe in this Figure that the long interdependency algorithm

performs better than the short one. More virtual bandwidth is made available of the same substrates and for the same sets of VN requests. It is especially relevant to see that the difference in performance is most noticed when the physical network as a smaller number of nodes. In this situation, when less mapping options are available and mapping is harder (which is noticeable by the small acceptance ratio for 15 substrate nodes on Figure 3.6(b)), it's when the long interdependency algorithm seems to be a better choice when compared to the short one. When the number of substrate nodes increases and reaches 45 the difference of performance gets smaller until there's almost no difference and the algorithms have very similar values for virtual bandwidth embedded and substrate bandwidth in use. On the other hand, we can also see that when the frequency of the VN requests increases in Figure 3.7(a), the performance of the algorithms exhibits a larger difference. This might be due to the fact that, as the frequency of the requests increases more VNs are accepted and the network becomes more saturated, making some links unavailable. This will allow the long interdependency algorithm to find solutions where the short interdependency algorithm wouldn't be able to find.

The same kind of behavior which was described for the bandwidth ratio can be observed in Figure 3.8, regarding the RAM ratio, and in Figure 3.9, regarding the VM ratio. The RAM ratio is measured as the time average RAM allocated to virtual resources over the total RAM of the substrate. The VM ratio is measured as the time average of the number of virtual machines in each physical node.

We have also tested what happens when we change the amount of links in the physical substrate offering high-security properties and the amount of virtual links requiring high-security. In Figure 3.10(a) we can see how having the same percentage of security links in the virtual and physical networks, while changing that percentage simultaneously for both of them, affects the VN acceptance ratio. On Figure 3.10(b) we can see what happens when substrate links with security remain at 30% and virtual links with security go from 10% to 50%.

Observing Figure 3.10(a) we can see that, as the number of these links increases in both physical and VNs, the acceptance ratio decreases and then increases again. This phenomenon is due to the initial difficulty in finding physical paths for the growing number of virtual links with security when there are few physical security links, but it is alleviated when the number of high-security links in the substrate network rises to significant levels. The long interdependency algorithm performs better, especially when the number of security links in both physical and VNs increases. It should be noted that even when none of the links requires or offers security, therefore rendering the security factor as if it was not present, the improvement in the algorithm performance is significant. This is due to the consideration of other limiting characteristics of the links such as the available bandwidth.

On the other hand, in Figure 3.10(b) we can see that when we set the physical amount of security links and increase the number of virtual links with high-security requirements, the acceptance ratio falls. This is expectable due to the few links available to host the ever growing number of virtual security links.

These behaviors are also evident in Figures 3.11, 3.12 and 3.13.

We took also a look in more detail to the Scenario 1, where the number of physical nodes is set at 35, the frequency of VN requests is set at 2 per time unit and 30% of the links in physical and VNs are deemed as high-security links. In Figure 3.14 one can see how the number of nodes of a VN influences the likelihood that the VN is accepted and the corresponding mapping time.

We can see that VNs with more nodes are less likely to be embedded and take more time to be mapped. The evolution of the times is not linear, and there is a decrease in the growth of the mapping times, which is possibly due to the fact that larger networks are less likely to be successfully mapped and therefore take less time being mapped than if they succeeded.

The long interdependency algorithm is able to embed more VNs but it takes longer time to map. So we have a trade-off between the quality of the algorithm and the time it takes to perform. Since the mapping time can be greatly reduced by running the algorithm in a machine with higher performance, it seems to me that this should not prevent the operator from using an algorithm that will allow him to map more VNs and virtual machines while using the same physical infrastructure, therefore increasing significantly the revenues of selling those virtual resources while maintaining or slightly increasing the costs for the physical infrastructure.

3.3 VN Mapping - Cost calculation for node and link placement

In [41] a heuristic algorithm based on cost calculation to choose the best node and link placement is presented and some results are discussed. In this section we take the algorithm proposed in the previous section, which is based in [41] but considers long interdependency, and we observe what happens when we change the ways of calculating node and link stress with the aim of maximizing the amount of virtual resources that it is possible to embed in different substrate networks. In the following sections we will present new formulas together with comparative results from the simulator tests and a brief analysis of these results. The parameters used are the same as in Scenario 1, except that we consider that all links have the same security characteristics.

3.3.1 Node Stress - Considering the CPU frequency

In [41] the node stress formula is written so that nodes with higher CPU frequencies will have lower stress values:

$$S_N = \frac{\text{Number of Active VMs}}{\delta + \text{Free RAM} \cdot \text{CPU Freq} \cdot (\text{N.CPU - Load})} \tag{3.1}$$

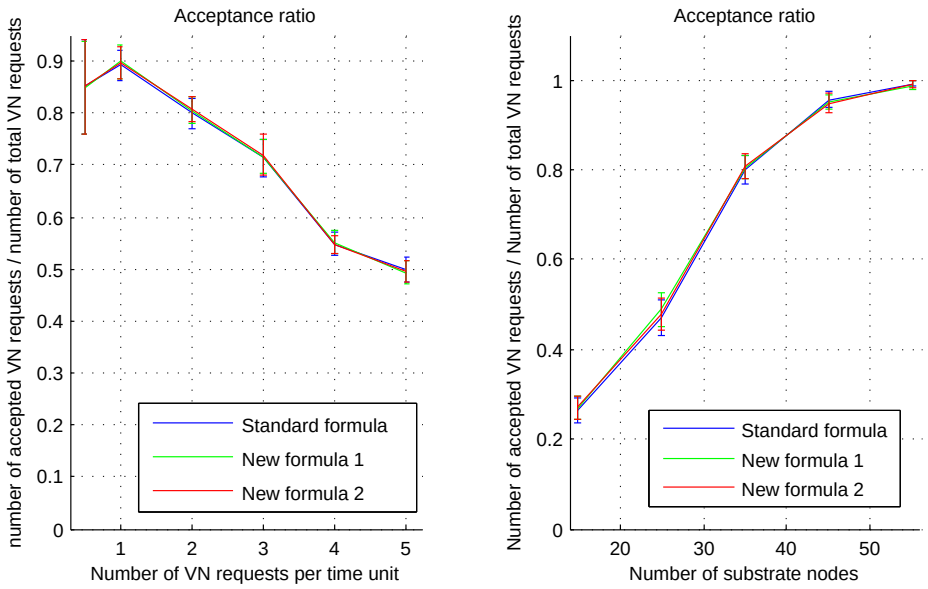
This option was made based on the consideration that nodes with higher CPU frequencies have more computer power and thus should be able to host more virtual nodes. With a lower stress value as the CPU frequency increases, these nodes will be more prone to be selected compared to nodes with lower CPU frequencies. We argue that, since a physical node can only host virtual nodes with equal or lower CPU frequencies than its own CPU frequency, it is the physical nodes with lower CPU frequencies that should be more prone to be selected, since they can only be candidates to a small part of the virtual nodes. This way, we would prevent physical nodes with lower CPU frequencies from being underused and those with higher CPU frequencies from being overwhelmed with virtual nodes of all frequencies and possibly so full that they cannot accommodate virtual nodes with high CPU frequencies that can only be hosted by these nodes. In order to achieve this we propose that CPU frequency is not considered when calculating the node stress, or considered in a way that lowers the

stress of nodes with lower CPU frequencies, as shown in the following node stress formulas:

$$S_N = \frac{\text{Number of Active VMs}}{\delta + \text{Free RAM} \cdot (\text{N.CPU - Load})} \tag{3.2}$$

$$S_N = \frac{\text{Number of Active VMs} \cdot \text{CPU Freq}}{\delta + \text{Free RAM} \cdot (\text{N.CPU - Load})} \tag{3.3}$$

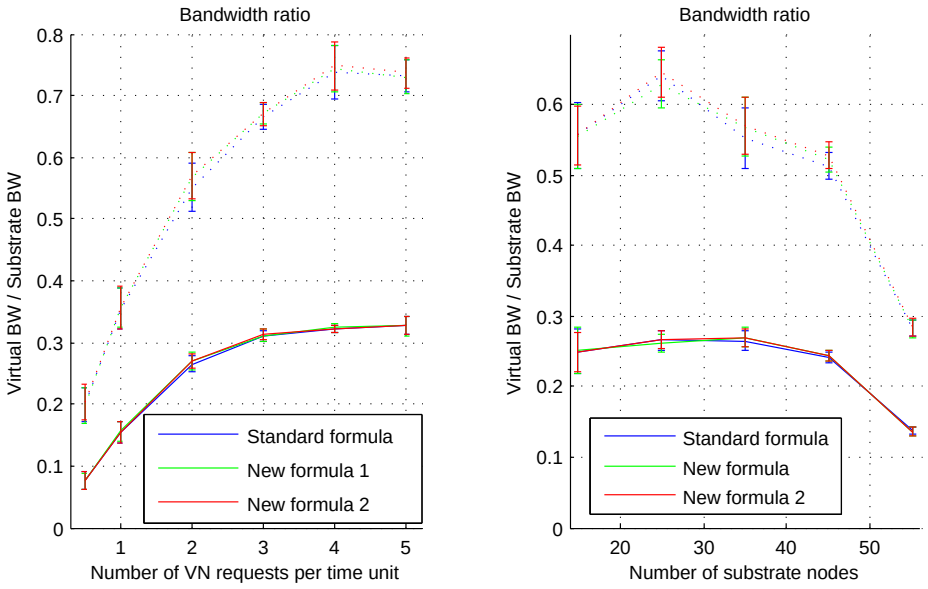
3.3.2 Simulation Tests and Results



(a) Acceptance Ratio vs. VN Request Rate

(b) Acceptance Ratio vs. Number of Substrate Nodes

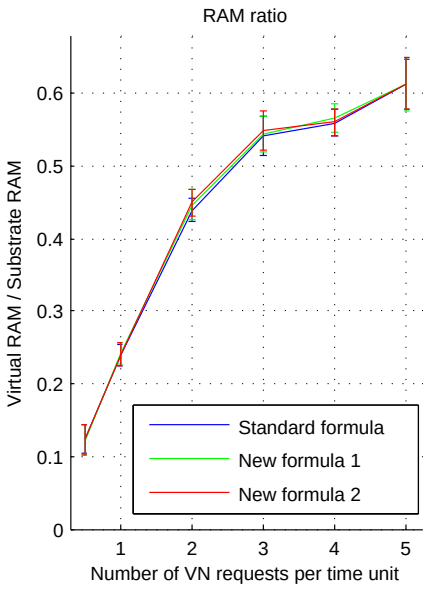
Figure 3.15: Rate of VNs requests accepted for different Node Stress formulas - use of the CPU Freq parameter



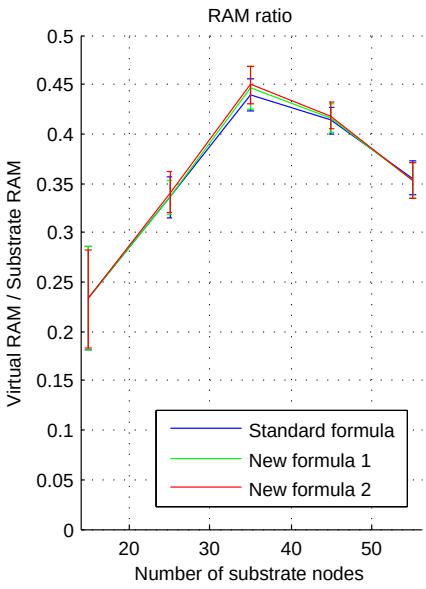
(a) Bandwidth Ratio vs. VN Request Rate

(b) Bandwidth Ratio vs. Number of Substrate Nodes

Figure 3.16: Ratio of Virtual BW in use over Substrate BW for different Node Stress formulas - use of the CPU Freq parameter

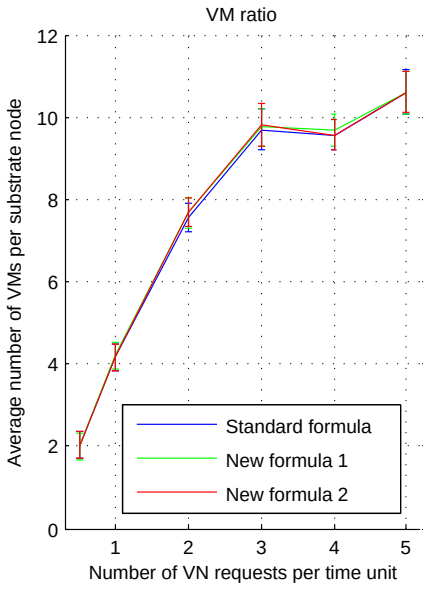


(a) RAM Ratio vs. VN Request Rate

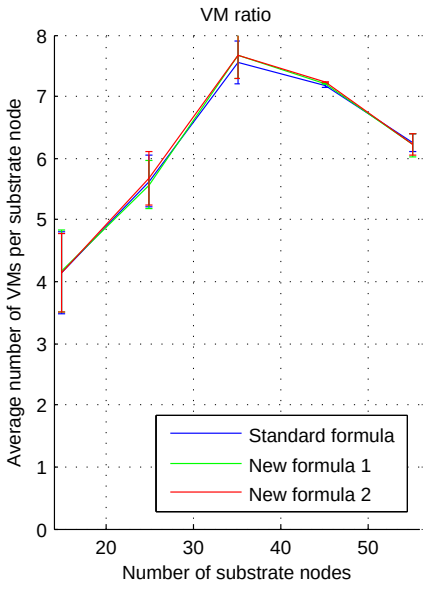


(b) RAM Ratio vs. Number of Substrate Nodes

Figure 3.17: Ratio of RAM in use over Total Substrate RAM for different Node Stress formulas - use of the CPU Freq parameter



(a) VM Ratio vs. VN Request Rate



(b) VM Ratio vs. Number of Substrate Nodes

Figure 3.18: Average Number of VMs per Substrate Node for different Node Stress formulas - use of the CPU Freq parameter

3.3.3 Discussion

Using the discrete event simulator to compare the use of the different formulas regarding the use of the CPU Freq parameter to calculate the node stress, the results shown in figures 3.15, 3.16, 3.17 and 3.18 were obtained. Analyzing the figures, one cannot conclude which formula is the best, as none of them is consistently nor significantly better than the others. What this might indicate is that the use of the CPU Freq parameter in the formula is irrelevant, so the parameter might be omitted in order to have a simpler formula for node stress calculation, as it happens with one of the evaluated formulas.

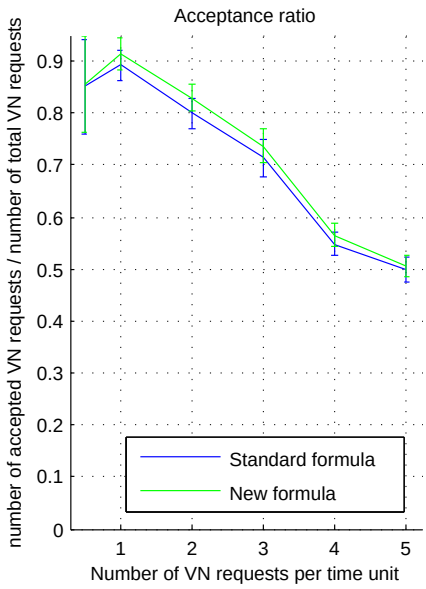
3.3.4 Node Stress - Considering available RAM and Load

RAM and Load are the parameters in the stress node formula that represent an amount of a limited resource that is progressively occupied. In equation 3.1 the stress of the physical nodes is inversely proportional to the amount of free Load and RAM. This option is based on the consideration that a network with a balanced amount of free resources in the nodes will be able to maximize the amount of virtual resources it can map and embed. We argue that this is a good approach to balance the amount of free resources in the nodes, but it will have as trade-off longer physical paths for the virtual links between the nodes (since more distant nodes might be chosen if they have more free resources). Since virtual nodes always occupy the same amount of physical resources in any physical node, special attention should be payed to the amount of physical links used to map the virtual links, since one might save bandwidth by using shorter paths and, as stated in [41], the bandwidth of the physical links represents the main limiting constraint to the amount of embeddable virtual resources. In order to prevent long physical paths from being used when the occupation of the nodes is not at a critical level, we propose that node stress is calculated as in equation 3.4.

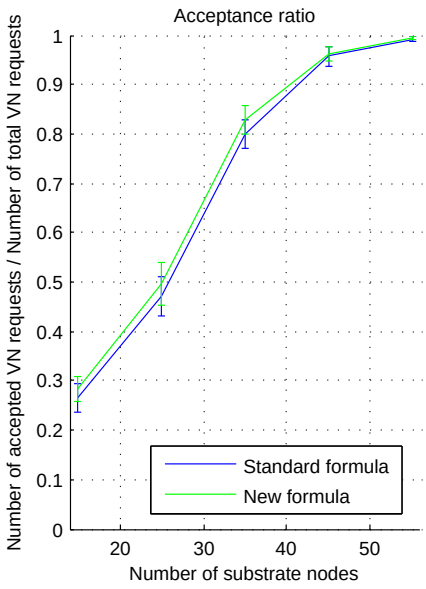
$$S_N = \frac{\text{Number of Active VMs}}{\text{CPU Freq}} \cdot (1 + \frac{k \cdot Load_{med}}{\text{N.CPU - Load}}) \cdot (1 + \frac{k \cdot RAM_{medReq}}{RAM_{free}}) \tag{3.4}$$

In this equation, RAM_{medReq} represents the average RAM of the virtual nodes and $Load_{med}$ represents the average Load increase for each virtual node embedded. k represents a constant value. This way, when calculating the potential of the candidate nodes (which is a product of the link cost and node stress) , link cost will be the most important parameter, until the considered nodes achieve a critical occupation state (that can be adjusted through constant k). In the simulations made we used k = 3.

3.3.5 Simulation Tests and Results

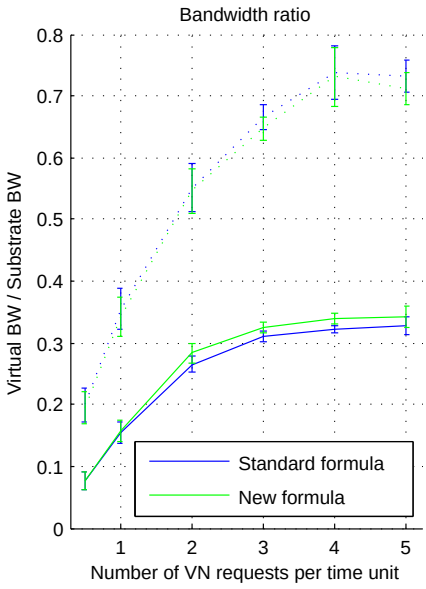


(a) Acceptance Ratio vs. VN Request Rate

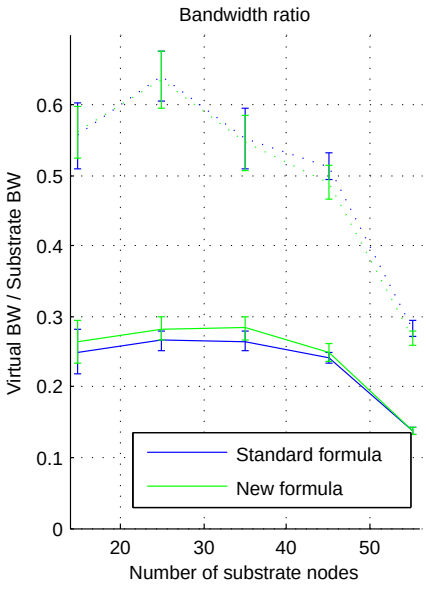


(b) Acceptance Ratio vs. Number of Substrate Nodes

Figure 3.19: Rate of VNs requests accepted for different Node Stress formulas - considering RAM and Load

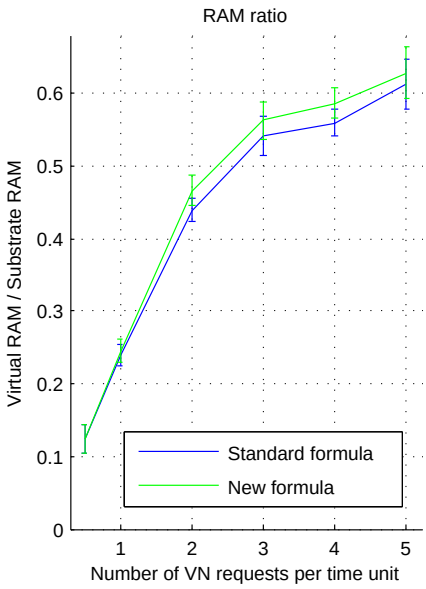


(a) Bandwidth Ratio vs. VN Request Rate

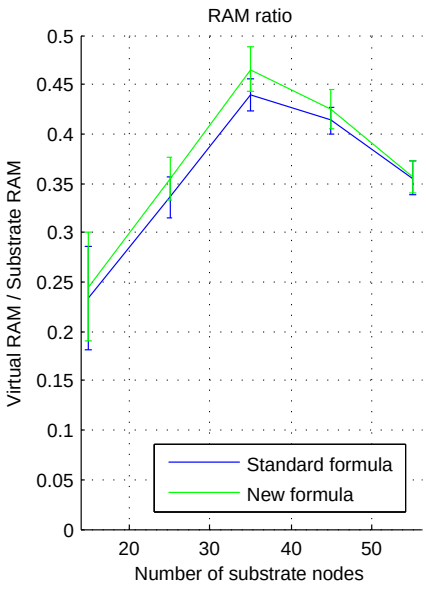


(b) Bandwidth Ratio vs. Number of Substrate Nodes

Figure 3.20: Ratio of Virtual BW in use over Substrate BW for different Node Stress formulas - considering RAM and Load

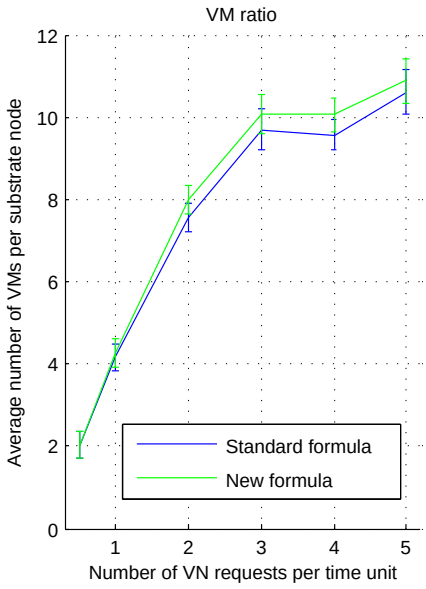


(a) RAM Ratio vs. VN Request Rate

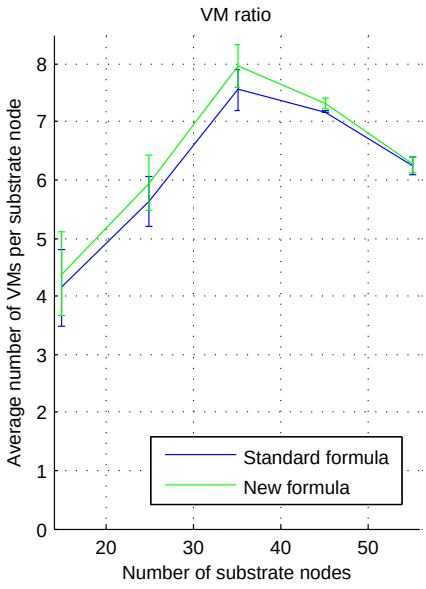


(b) RAM Ratio vs. Number of Substrate Nodes

Figure 3.21: Ratio of RAM in use over Total Substrate RAM for different Node Stress formulas - considering RAM and Load



(a) VM Ratio vs. VN Request Rate



(b) VM Ratio vs. Number of Substrate Nodes

Figure 3.22: Average Number of VMs per Substrate Node for different Node Stress formulas - considering RAM and Load

3.3.6 Discussion

Using the discrete event simulator to compare the use of the different formulas regarding the use of the RAM and Load parameters to calculate the node stress, the results shown in figures 3.19, 3.20, 3.21 and 3.22 were obtained. Analyzing the figures, one can see that the proposed new node stress formula slightly improved the performance of the mapping algorithm in all of the evaluated situations. It should be especially noted in figures 3.20(a) and 3.20(b) that the virtual bandwidth embedded using the new formula increases while the bandwidth occupied decreases, this means more revenue for less cost. It is also visible that the performance difference is larger when the number of physical nodes decreases and the frequency of the VN requests increases.

3.3.7 Link Stress - Considering available bandwidth

In [41] link stress is the value of the bandwidth in use in the physical link:

$$S_L = BW_{occupied} \tag{3.5}$$

This option is based on the consideration that all physical links have the same bandwidth (or this bandwidth is not known) and that physical links with more bandwidth in use should not be chosen. We argue that this option is good when trying to balance the link load, but it does not consider the heterogeneity of the physical links and it balances the amount of occupied bandwidth of the physical links at the expense of using more physical links, thus occupying more physical resources. We propose 2 formulas to address theses problems. In 3.6 we take into consideration the total amount of bandwidth of the physical link, while in 3.7 we consider that the link cost should be determined by the number of physical links occupied by the virtual link until the amount of available bandwidth in the physical links reaches a critical point. This critical point can be adjusted through the value of constant k. In the simulations made we used k = 3.

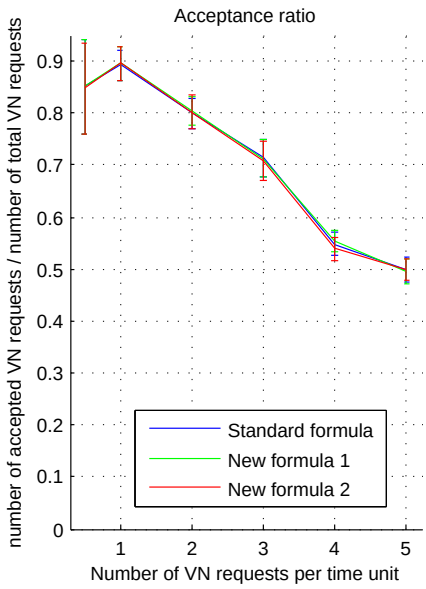
New Formula 1:

$$S_L = \frac{BW_{occupied}}{BW_{total}} \tag{3.6}$$

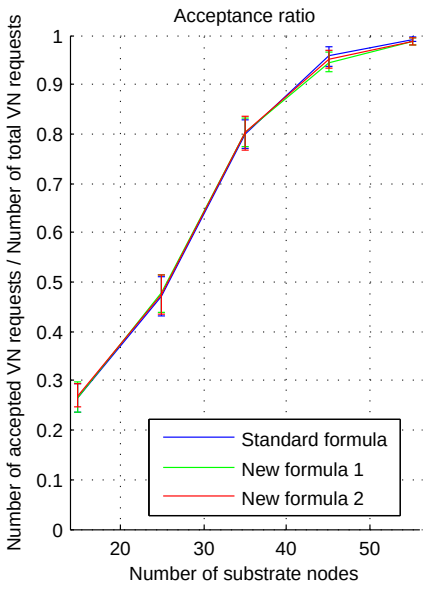
New Formula 2:

$$S_L = 1 + \frac{k \cdot BW_{medReq}}{BW_{free}} \tag{3.7}$$

3.3.8 Simulation Tests and Results

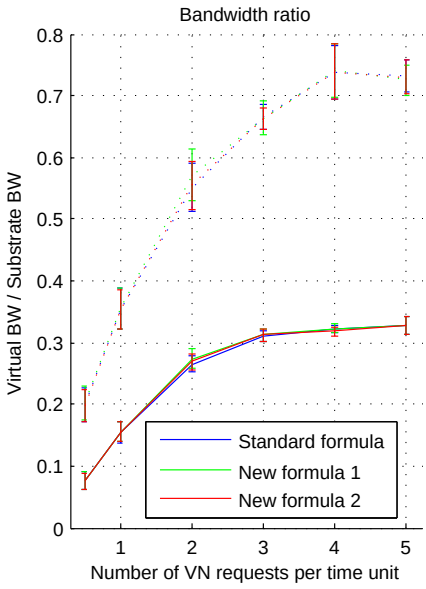


(a) Acceptance Ratio vs. VN Request Rate

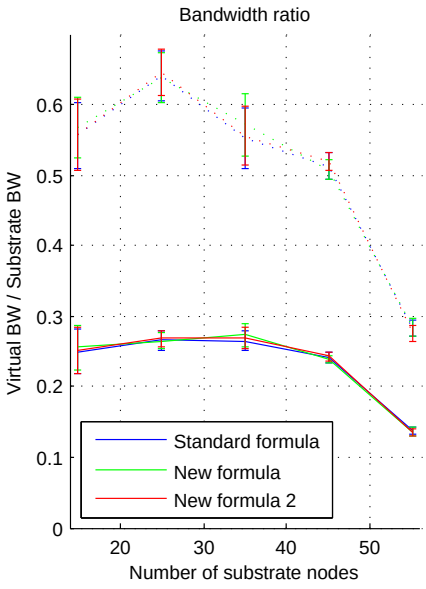


(b) Acceptance Ratio vs. Number of Substrate Nodes

Figure 3.23: Rate of VNs requests accepted for different Link Stress formulas

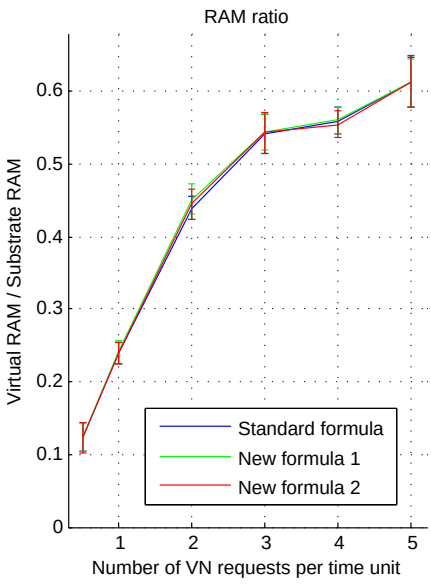


(a) Bandwidth Ratio vs. VN Request Rate

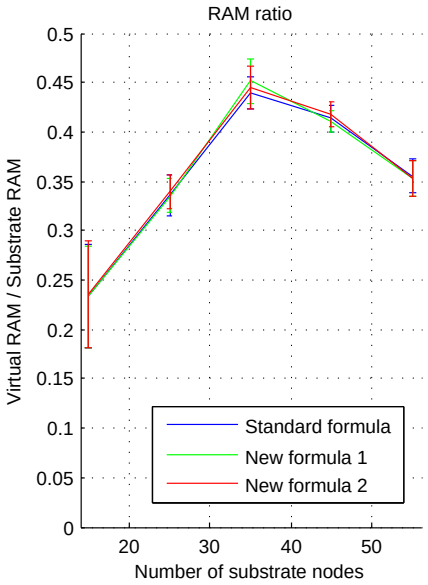


(b) Bandwidth Ratio vs. Number of Substrate Nodes

Figure 3.24: Ratio of Virtual BW in use over Substrate BW for different Link Stress formulas

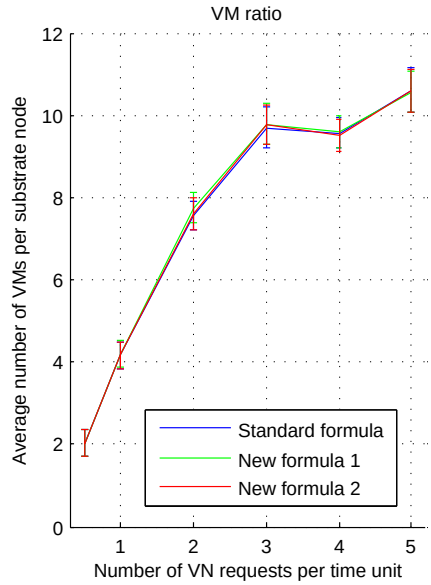


(a) RAM Ratio vs. VN Request Rate

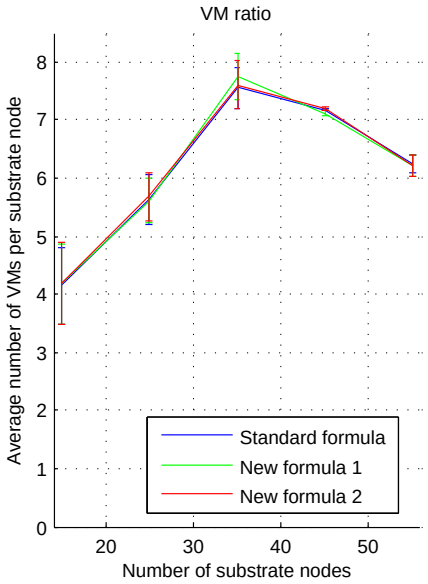


(b) RAM Ratio vs. Number of Substrate Nodes

Figure 3.25: Ratio of RAM in use over Total Substrate RAM for different Link Stress formulas



(a) VM Ratio vs. VN Request Rate



(b) VM Ratio vs. Number of Substrate Nodes

Figure 3.26: Average Number of VMs per Substrate Node for different Link Stress formulas

3.3.9 Discussion

Using the discrete event simulator to compare the use of the different formulas regarding the use of the bandwidth values of the physical links to calculate the link stress, the results

shown in figures 3.23, 3.24, 3.25 and 3.26 were obtained. Analyzing the figures, one can see that using any of the three formulas does not change much in the algorithm performance. This means that the only required information in the formula is the occupied bandwidth.

3.4 VN & Cloud Mapping

The mapping of cloud resources was not considered until this moment. We propose to take the algorithm proposed in [41] and develop it in order to consider also cloud resources. We will consider the cloud resources as being virtual servers. The main difference from cloud resources (servers) to the other resources (routing nodes) is that we consider cloud resources to also include HDD memory. We also consider CPU frequency not to be a limiting parameter when choosing the server host. This way, we now have 2 parameters, RAM and HDD memory, that are requested by the virtual servers and that only physical resources for cloud hosting with at least that amount of free resources can host. We consider that virtual servers are treated in the same way as virtual routers, with the differences that their candidates are restricted to physical servers (and virtual routers are restricted to physical routers) and their stress is calculated according to a different formula, which we named Server Stress. Server Stress is an indicator about how likely a physical server should be to host a virtual server in comparison with other servers. This indicator is used on the mapping algorithm to calculate the potential of a certain candidate server to host a virtual server. However, this potential is not calculated only using Server Stress but also considering the Link Stresses of the physical links which might be used to host virtual links if the virtual server is mapped on the physical server. This interplay between Server Stress and Link Stress (with the Link Stresses aggregated in the Link Cost indicator) allows for a node placement that considers both the servers' characteristics as well as the network characteristics.

We propose 3 formulas for the calculation of server stress: Formula 1, expressed in equation 3.8, has a similar structure to 3.1 and considers HDD memory the same way it considers the RAM; Formula 2, in equation 3.9, takes into account both the physical node available resources and the virtual node resource requests, trying to "fit" the virtual node in a physical node with a "profile" of free RAM and HDD memory resources similar to the virtual node requests; and Formula 3, eq. 3.10, considers HDD memory the same way that RAM and Load are considered in formula 3.4, with k = 3.

Formula 1:

$$S_S = \frac{\text{Number of Active VMs}}{\delta + \text{Free RAM} \cdot \text{Free HDD} \cdot (\text{N.CPU - Load})} \tag{3.8}$$

Formula 2:

$$S_S = \frac{\text{Number of Active VMs}}{\text{N.CPU - Load} + \delta} \cdot \frac{RAM_{free}}{RAM_{free} - RAM_{req} + \delta} \cdot \frac{HDD_{free}}{HDD_{free} - HDD_{req} + \delta} \tag{3.9}$$

Formula 3:

$$S_S = \text{Number of Active VMs} \cdot (1 + \frac{k \cdot Load_{med}}{\text{N.CPU - Load}}) \cdot (1 + \frac{k \cdot RAM_{medReq}}{RAM_{free}}) \cdot (1 + \frac{k \cdot HDD_{medReq}}{HDD_{free}}) \tag{3.10}$$

The characteristics of the virtual servers and routing nodes, link and physical servers and routing nodes are represented in tables 3.3, 3.4, 3.5 and 3.6. The characteristics of the virtual servers were based upon the virtual servers made available by Amazon.com [11].

N. CPUs	{2; 4; 6; 8}
CPU Frequency (GHz)	{2.0 to 3.2 in 0.2 steps }
RAM Memory (GB)	{2; 4; 6}
Link Bandwidth (Mbps)	{800; 1200}

Table 3.3: VN Mapping Simulation Scenario 2 - Physical Network Nodes and Links' parameters pool.

N. CPUs	{8; 16; 32; 64}
HDD Memory (GB)	{6400; 12800; 25600 }
RAM Memory (GB)	{256; 512; 1024}

Table 3.4: VN Mapping Simulation Scenario 2 - Physical Network Servers' parameters pool.

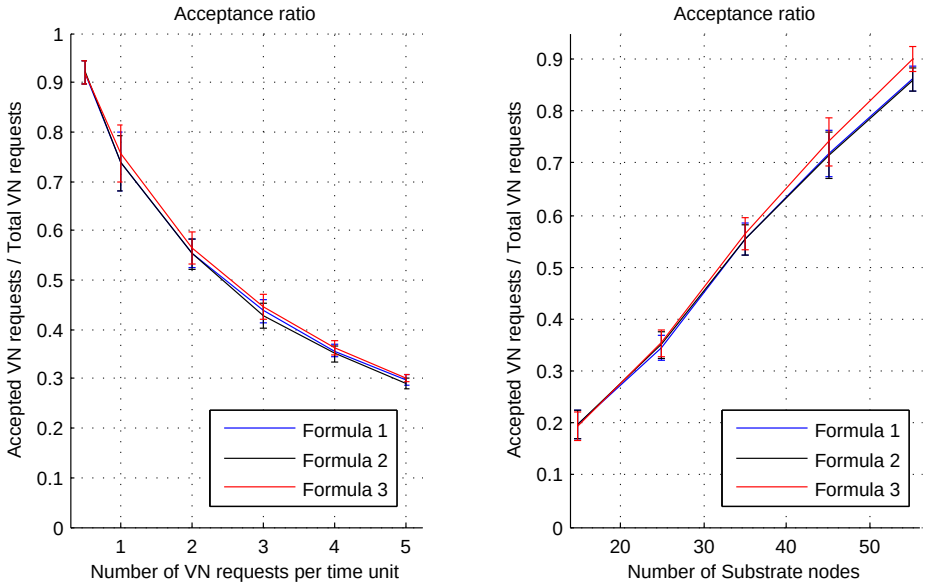
N. CPUs	{1; 2; 3; 4 }
CPU Frequency (GHz)	{2.0 to 3.2 in 0.1 steps }
RAM Memory (MB)	{64; 128; 256; 512 }
Link Bandwidth (Mbps)	{34.368 139.264 }

Table 3.5: VN Mapping Simulation Scenario 2 - VN Nodes and Links' parameters pool.

N. CPUs	{1; 2; 4; 8; 16; 32; 64 }
HDD Memory (GB)	{100; 200; 400; 800; 1600 }
RAM Memory (GB)	{2; 4; 8; 16; 32; 64 }

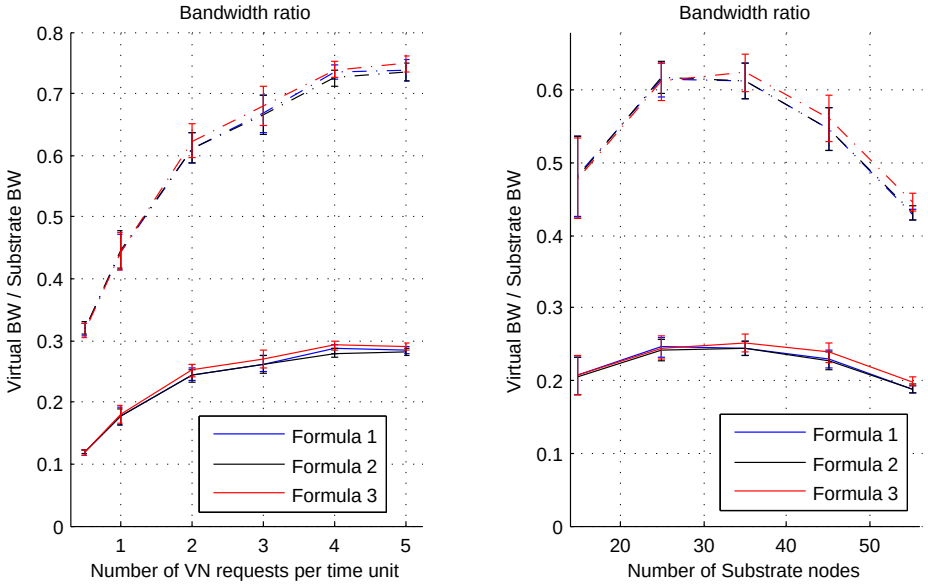
Table 3.6: VN Mapping Simulation Scenario 2 - VN Servers' parameters pool.

3.4.1 Simulation Tests and Results



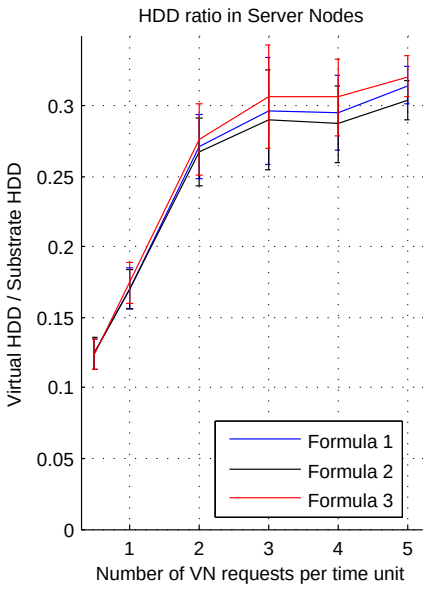
(a) Acceptance Ratio vs. VN Request Rate (b) Acceptance Ratio vs. Number of Substrate Nodes

Figure 3.27: Rate of VNs requests accepted for different Server Stress formulas

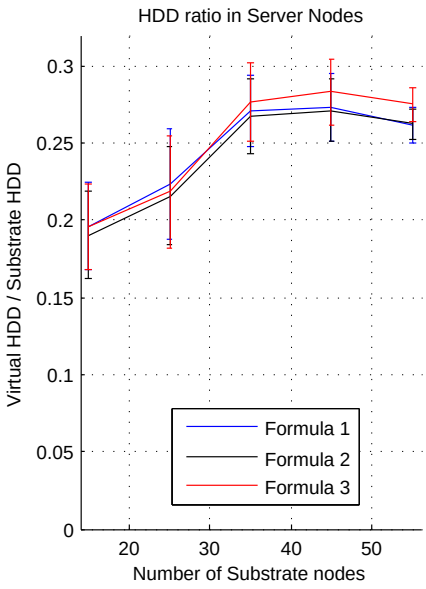


(a) Bandwidth Ratio vs. VN Request Rate (b) Bandwidth Ratio vs. Number of Substrate Nodes

Figure 3.28: Ratio of Virtual BW in use over Substrate BW for different Server Stress formulas

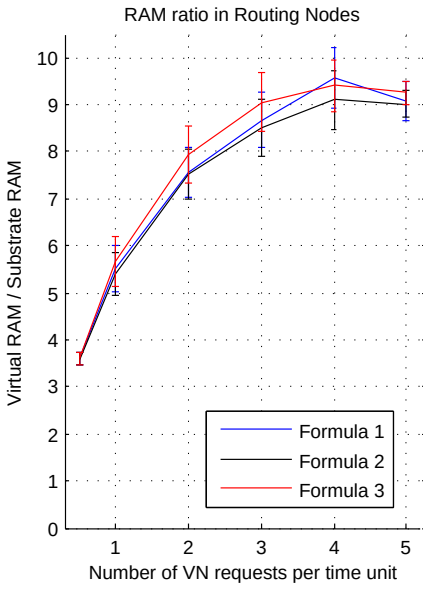


(a) HDD Ratio vs. VN Request Rate

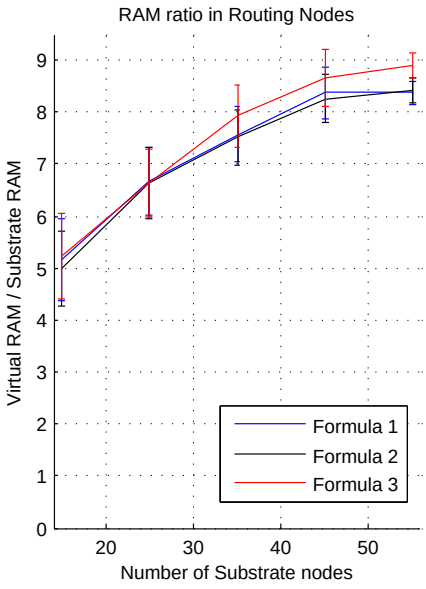


(b) HDD Ratio vs. Number of Substrate Nodes

Figure 3.29: Ratio of HDD in use over Total Substrate HDD for different Server Stress formulas

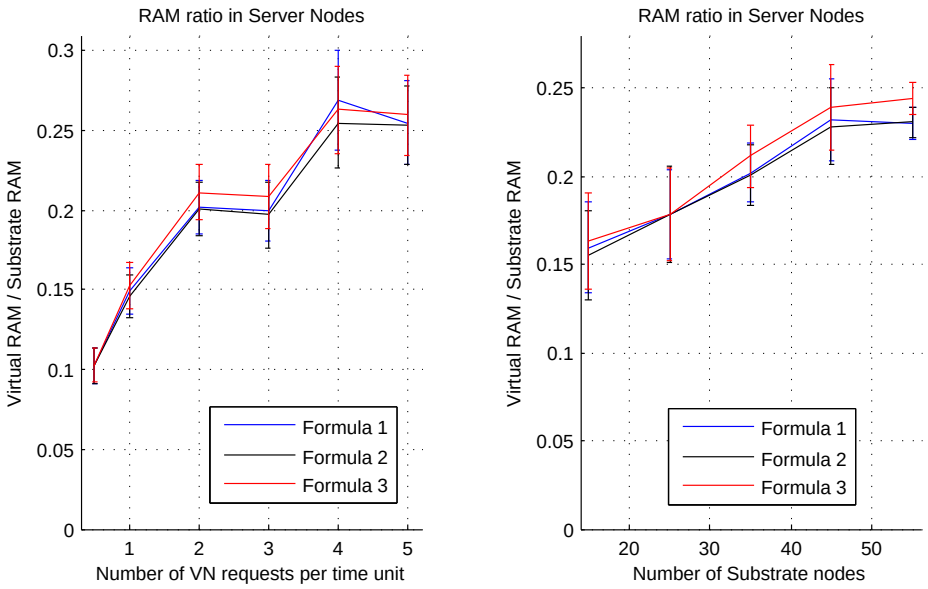


(a) RAM Ratio in Routing Nodes vs. VN Request Rate



(b) RAM Ratio in Routing Nodes vs. Number of Substrate Nodes

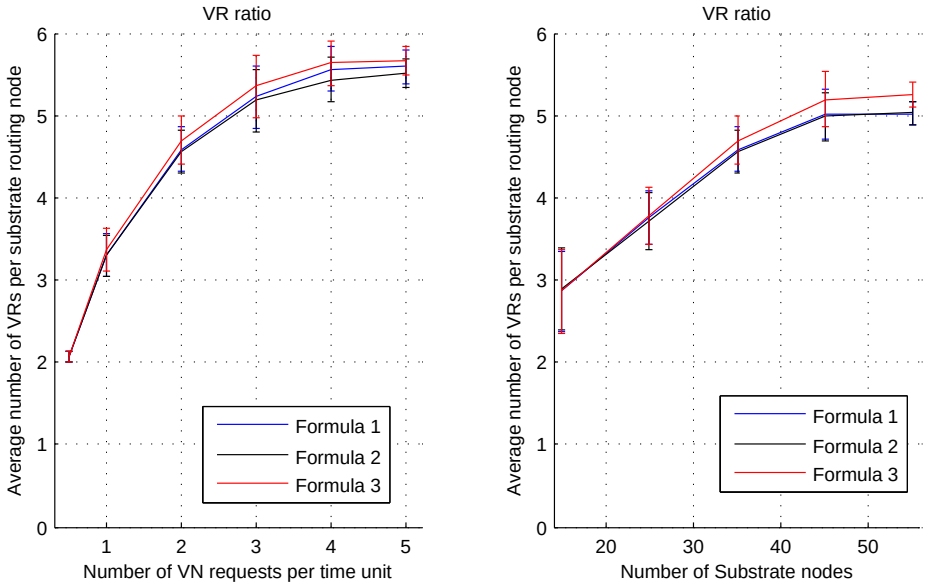
Figure 3.30: Ratio of RAM in use in Routing Nodes over Total Substrate RAM in Routing Nodes for different Server Stress formulas



(a) RAM Ratio in Server Nodes vs. VN Request Rate

(b) RAM Ratio in Server Nodes vs. Number of Substrate Nodes

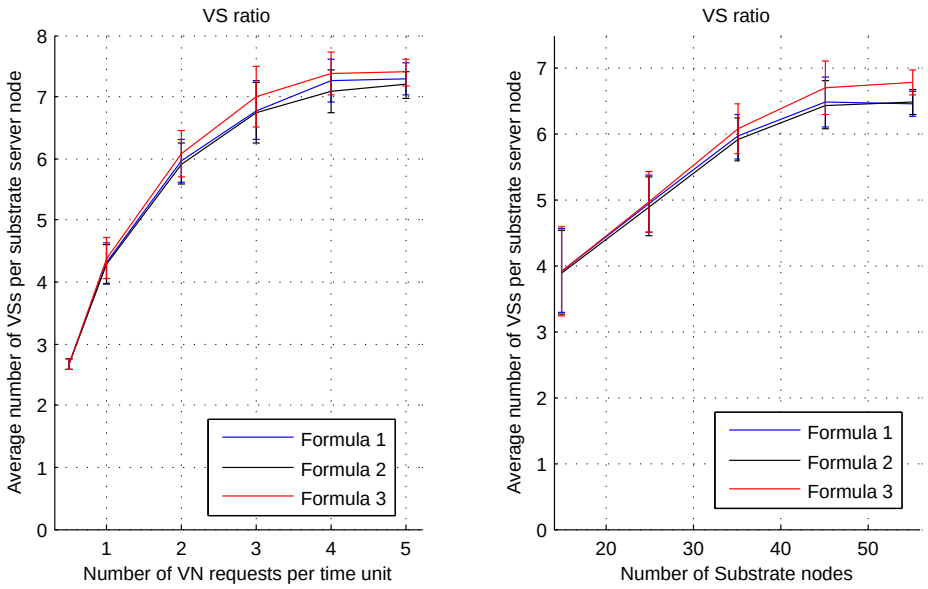
Figure 3.31: Ratio of RAM in use in Server Nodes over Total Substrate RAM in Server Nodes for different Server Stress formulas



(a) VR Ratio vs. VN Request Rate

(b) VR Ratio vs. Number of Substrate Nodes

Figure 3.32: Average Number of VRs per Substrate Routing Node for different Server Stress formulas



(a) VS Ratio vs. VN Request Rate

(b) VS Ratio vs. Number of Substrate Nodes

Figure 3.33: Average Number of VSs per Substrate Server Node for different Server Stress formulas

3.4.2 Discussion

Observing the results for the simulations with different frequencies of VN requests and different numbers of substrate nodes in Figures 3.27, 3.28, 3.29, 3.30, 3.31, 3.32, 3.33, to compare the different Server Stress formulas, one can see that the performance difference is small. In spite of this, it's visible that Formula 3, of eq. 3.10, performs better than the other formulas in almost all evaluated parameters and scenarios. It can be seen that the difference grows larger as the frequency of the VN requests increases and particularly as the number of substrate nodes increases. For example, when the number of substrate nodes is 55 (44 routing nodes and 11 server nodes), the acceptance ratio for Formula 3 is 90%, while for the other Formulas is about 86%. Analyzing this behavior it is expectable that for larger numbers of substrate nodes and/or higher VN request frequencies the performance difference will increase even more, thus making Formula 3 the best way of calculating the Server Stress.

3.5 Conclusions

The more virtual resources an operator can embed in his network, and thus sell to the client that requests a VN, the more revenue it will have. This way, algorithms that allow the embedding of a larger amount of virtual resources, even if at the cost of a worse load balancing, are more profitable, and therefore better algorithms from a commercial point of view.

After evaluating the algorithms as well as the different stress formulas proposed, there are some conclusions we can take. First, the extension of the mapping algorithm to consider

the long interdependency in node placement instead of just the neighbors placement (short interdependency) succeeds at enhancing the quality of the mapping decisions, allowing for more networks to be mapped and embedded successfully. However, it was also possible to see that this comes at a cost of a longer mapping time. Second, the CPU Freq. parameter may not be used to calculate node stress, as its removal from the stress formula does not affect the global mapping outcome. Third, it was observable that the new formula proposed to calculate node stress regarding Load and RAM performs better in all scenarios, which indicates that it should be adopted. Fourth, the new formulas proposed to calculate link stress do not change the global outcome when compared to the one already in use in the NVSS. Since the current formula needs less information and it is simpler than the other formulas, it should be kept.

As for the mapping of cloud resources it seems that, of the 3 different server stress formulas proposed, the best one is Formula 3. This formula considers the average characteristics requests of the virtual servers and uses a non-linear approach to model the relation between node occupation and node stress.

The consideration that stress should increase more as the remaining resources approach a critical level, while not changing much when the is a lot of free resources, seems to be a successful approach both for Node Stress as for Server Stress.

Chapter 4

Platform Requirements Specification

4.1 Introduction

In the State of the Art chapter we presented the network virtualization platform NVSS. This VN platform possesses four main features: physical and VN and resource discovery, physical and VN monitoring, VN deployment and mapping in the physical substrate, and management of VNs.

These features provide a solid basis over which advanced mechanisms for virtual networking can be developed. The new mechanisms that are presented in this chapter were conceived considering that the operator aim is to maximize the revenue and reduce costs. This can be achieved through the optimization and consolidation of the already implemented features, as in the improvement of the mapping algorithms, and through the inclusion of new features that offer new solutions for the clients.

The resulting platform contains tools for the operators to provide two types of solutions: a VN solution where the complete VN, which might include cloud resources, is designed by the client, and a solution where cloud resources and connectivity between sites are requested and provided without the client having to design the complete network, similarly to a VPN. Tools to manage physical and virtual cloud resources are also added, so that the platform can distinguish the different types of resources. This includes new mapping mechanisms which perform integrated mapping of both network and cloud.

The aim of this chapter is to provide an overview of the main advanced mechanisms that shall be added to the platform. It also introduces the intended uses, interfaces and performance of the platform. The base features of the platform, the user classes, operating environment, constraints and dependencies, and the performance, security and software quality requirements have already been presented in the State of the Art chapter and will not be presented here since they remain mostly unchanged.

4.2 Advanced Mechanisms Overall Description

The main advanced mechanisms added to the NVSS platform can be described in two parts.

The first of them is CC. The base features of the NVSS included the design, mapping and enforcement of VNs containing virtual routers as nodes but not servers. We propose

an algorithm for server mapping, and offer the possibility to request, map and create virtual servers.

The second advanced mechanism is providing connectivity, which is a feature that allows the user to specify what he needs in terms of connectivity by designing a hose model of the network. In a hose model the network is defined by the amount of bandwidth that each resource can send to the network and receive from it. It does not specify a topology. In the end, the user will get a VN that respects the hose model that was specified and the cloud resources that were requested, in case they were requested.

4.3 System Advanced Mechanisms Details

4.3.1 VN & Cloud

CC is one of the hot areas of research and business investment nowadays. Server virtualization has enabled infrastructure operators to offer IT resources to enterprises and clients who wish to outsource them in a way that creates a great reduction of infrastructure costs by sharing common infrastructure. This way, it is significant that a platform for VN and resource management can also deal with virtual and physical servers for CC.

This feature allows the NVSS to discover which physical nodes are servers and to consider them in the graphical interface, both for the design of the VN as for the discovery of the physical and virtual resources. It also takes into account the HDD information of both virtual and physical servers in order to optimize the mapping of the virtual servers, according to the algorithm that was proposed in the Mapping Algorithms chapter.

This way, the platform allows the design and creation of VNs together with Cloud resources.

4.3.2 VPN & Cloud

With this platform we have the possibility to design and enforce a complete VN. This solution is attractive to clients who need high-performing network services or who wish to have a great degree of control over the network. However, this is just too much for clients who only wish to have connectivity between their sites and possibly to use cloud resources.

This feature allows the client who wishes to have connectivity and possibly to use cloud resources to select the sites which he wants to connect and the characteristics of the cloud resources he wishes to have available. Instead of designing the complete network topology the client just needs to select the bandwidth requirements from and to each site and cloud resource according to a hose-model. This will allow the operator to act on two different markets for network virtualization: on the one hand it will be able to provide full-blown VNs with cloud resources to clients who wish to provide high-performing services running over the VN and/or have a great degree of control over the network. This is a service which is attractive for companies who run very specific services over the network, such as telephone companies, for example. On the other hand the operator will be able to offer an integrated solution to clients who require connectivity and cloud resources but don't need or wish to design the complete network. These clients can request the VN through an easier mechanism and will still have full control over the VN created. Meanwhile, the operator can reduce costs by optimizing, according to its own needs, the network design phase. This

solution is attractive for clients for whom the topology of the VN is not a critical aspect but still wish to have a great degree of control over the network.

This way, the operator will have an easy to use, automatic and optimized tool to reach out to clients who don't care about the topology of the network, as long as the connectivity and cloud resource requirements are being respected.

4.4 Interface Requirements

In this section, the main user interactions will be analyzed, and so will the software's communication interfaces and semantics.

4.4.1 Use cases

Several uses cases, shown in figures 4.1 and 4.2, can be considered. The 'Create VNet' action can be performed by an user in order to create a new VN with or without cloud resources. As an alternative, the Create VPN action can be used in order to request a VPN style network with or without Cloud resources. It is also possible to select a previously built XML containing the description of a VN or a VPN which can be loaded resorting to the Load VNet XML and Load VPN XML actions. After performing any one of these actions, a created or loaded VN or VPN may be selected and additional actions may be performed to further specify and configure it. The available actions can be observed in figures 4.1 and 4.2.

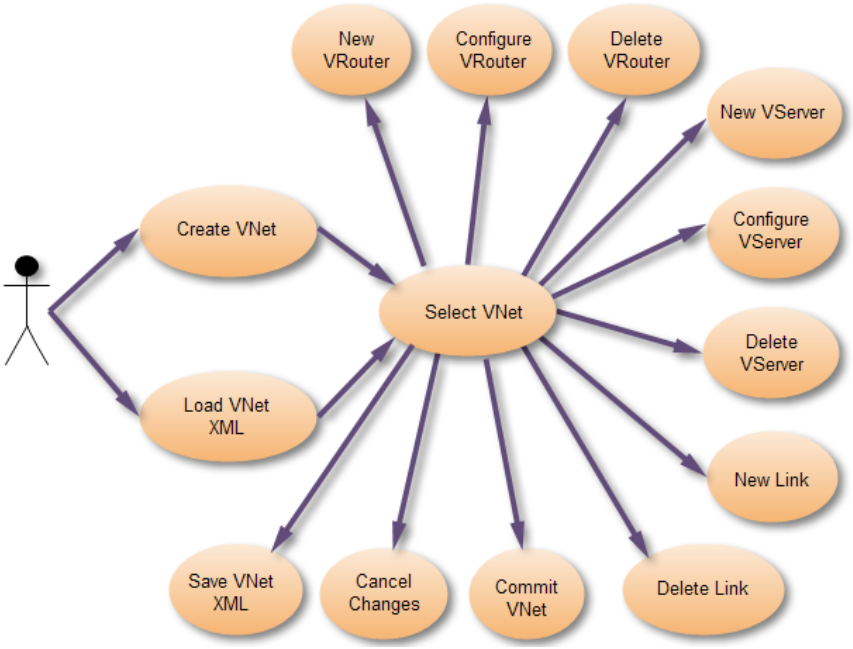


Figure 4.1: Simplified VNet Creation and use-cases

In figure 4.2, different use cases are considered. One such example is the Get VNet/VPN action which triggers a request for a VN or VPN that can be afterwards managed or monitored. Multiple requests for different VNs may be performed. The Manage VNet action allows the user to delete the related VN or VPN, through the Delete VNet/VPN action, or

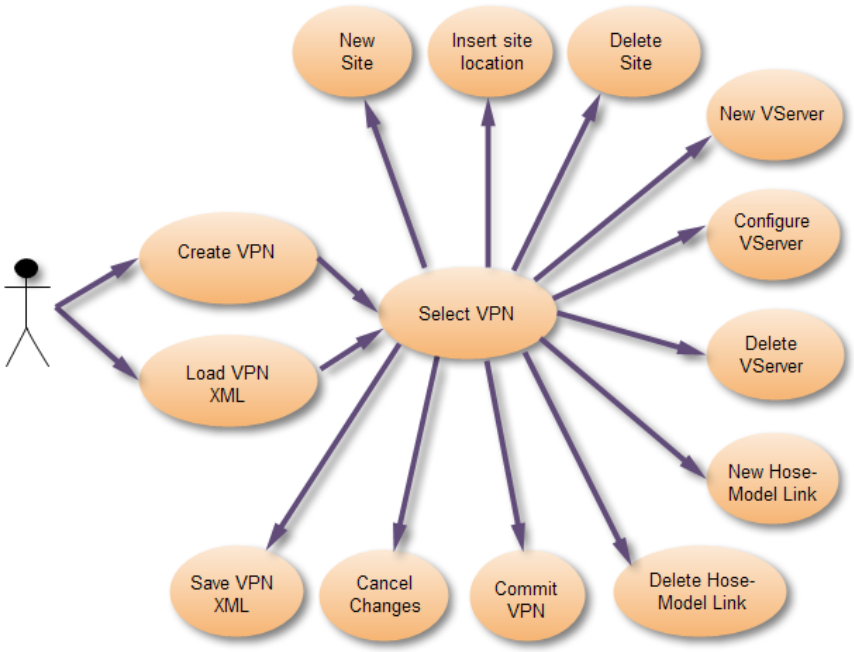


Figure 4.2: Simplified VPN Creation and use-cases

to modify some parameters of a given resource such as its state, HDD memory and RAM by using the Modify Resource action. Monitoring actions are also available through the Monitor VNet/VPN action that allows the user to View Resource Properties.

4.4.2 User interface

Main Menus

In order to easily allow the user to perform the previously specified actions, a user interface is provided. The user interface provides the so-called *coolbar* with buttons whose functions can be easily identified by their icons. They perform VN and VPN build tasks, which are different as the user is configuring a VN or a VPN. Their function is, according to the notation "VN function / VPN function" when different:

- New Virtual Router / New Site;
- New Virtual Server;
- New Link / New Hose Model Link;
- Delete Link / Delete Hose Model Link;
- Delete Resource;
- Commit VN / Commit VPN;
- Cancel Modifications;

Besides these buttons, a menu bar is also provided, in the *Actions* menu the following actions may be performed:

- *Create* a new VN;
- *Create* a new VPN;
- *Save* the current VN or VPN to an Extensible Markup Language (XML) file;
- *Load* a VN or VPN from an XML file;
- *Quit*;

There is also a *Get VNet* menu which will trigger a drop-down menu with the existing VNs and VPNs. Upon selecting one VN, a new tab will appear with the VN and subsequent updates to that VN will be reflected in the Graphical User Interface (GUI).

The last menu is the *Help* menu, which provides some information about the GUI.

Context Menus

By right-clicking in any one of the resource, a context menu will appear with the related options. The context menus will depend on the resource type and will provide information about the resources' configuration and actions that can be executed.

4.5 Conclusions

This chapter described the main advanced mechanisms for the platform, the intended uses for them, and the graphical and software interfaces. New advanced mechanisms were proposed such as VN & CC and VPN & CC and the considered use-cases explained. This serves as a reference for the development of the platform along the rest of this Thesis.

Chapter 5

Software Implementation

5.1 Introduction

The aim of this chapter is to provide insight about the mechanisms through which the features presented were added to the platform. In section 5.2 the main databases, classes and structures of the 3 modules of the platform will be presented. In the following sections it will be described how the different physical resources are identified and how VNs and VPNs are configured, designed and mapped.

5.2 Main Databases, Classes and Structures

5.2.1 Control Center

There are six main databases in the Control Center modules (5.1), three containing display objects and three other containing the resources', links' and virtual networks' information.

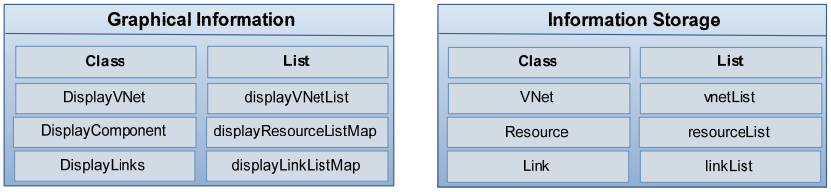
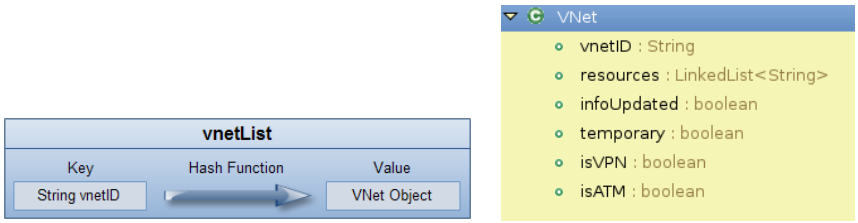


Figure 5.1: Control Center's Classes and Lists.

The *VNet database*, called *vnetList*, uses a hash map to store its keys and values (figure 5.2(a)). The keys are the unique VNet ID's while the values are VNet Objects, represented on the class diagram of figure 5.2(b).

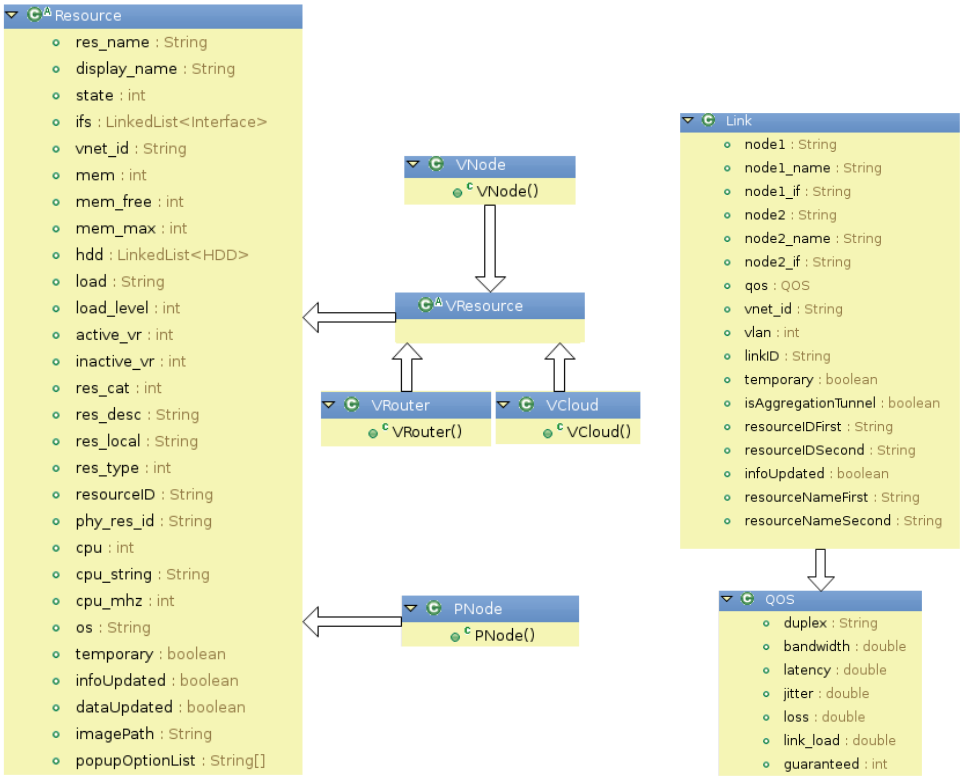
The *Resource database*, implemented as a *resourceList*, is also based on hash maps. Its *Resource* objects store data according to the class diagram of figure 5.3(a). There can be two main types of resources, physical (*PNode*) and virtual (*VResource*), and the latter can have three types, either virtual node (*VNode*), virtual router (*VRouter*) or virtual cloud (*VCloud*).

The *Link database* also uses a hash map to store its link objects (figure 5.3(b)).



(a) VNet list hash map. (b) VNet class.

Figure 5.2: Control Center's VNet list hash map and class.



(a) Resource class. (b) Link class

Figure 5.3: Control Center's Resource and Link classes.

The remaining three databases store display objects. Starting with the *Display VNet database*, its database is called *displayVNetList* and the hash map is similar to the previous ones: the VNet ID is the key while the value is the DisplayVNet object.

The *DisplayComponent* database uses a double hash map to store its data, the first key is a VNetID. It leads to another hash map whose key is now the resource ID and the value is a DisplayComponent object.

Finally, the *Display Link database* utilizes a hash map whose key is a VNet ID that leads to a linked list of DisplayLink objects. The Display objects are described by the classes shown in figure 5.4

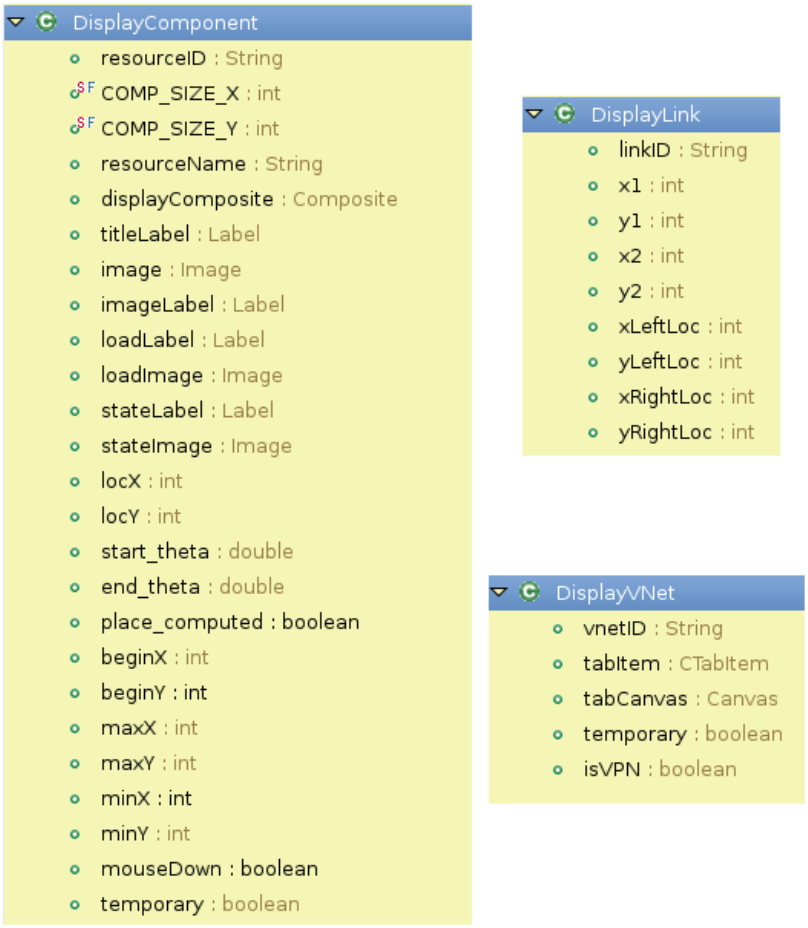


Figure 5.4: Control Center's Display classes.

5.2.2 Manager

Main Databases

The Manager uses linked lists to store all its data. Regarding the resources', links' and virtual networks' data storage, there are 2 main databases: the *VNet List* is a linked list of pointers to VNet structures, each one containing a linked-list of pointers to *VNet Nodes* which in turn also have a linked list of pointers to related *Links*. This hierarchical

architecture optimizes resource and link searches. The full data structure hierarchies can be seen on figure 5.5

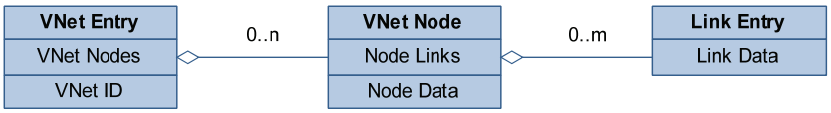


Figure 5.5: Manager's VNet Entry.

The second database is a redundant one, the *Main Resource List* is a linked list that holds pointers to all the resources. This pointer database is kept for compatibility purposes.

Each database has its own associated mutex, used for synchronizing access from multiple threads.

Auxiliary Databases

Since the Manager accepts connections to multiple Control Centers and Agents, the storage of information regarding the connected modules is required. To that end, additional linked lists exist that store the Agents' and Control Centers' information. These connection entries are depicted in figure 5.6.

The connected Agents' data structure contains the necessary socket ID, physical resource ID, information about the Agent's connection handler thread and the time of last contact.

The data structure containing the Control Centers' connection information is a bit more complex. It contains the Control Center's ID, the requested VNets, a temporary list and an internal message linked list used for sending messages to the respective Control Center. Thread synchronization and control variables are also included within this data structure.

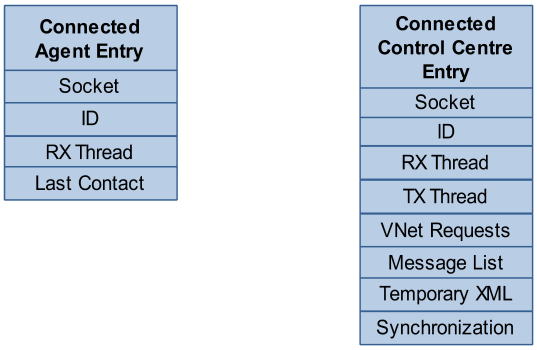
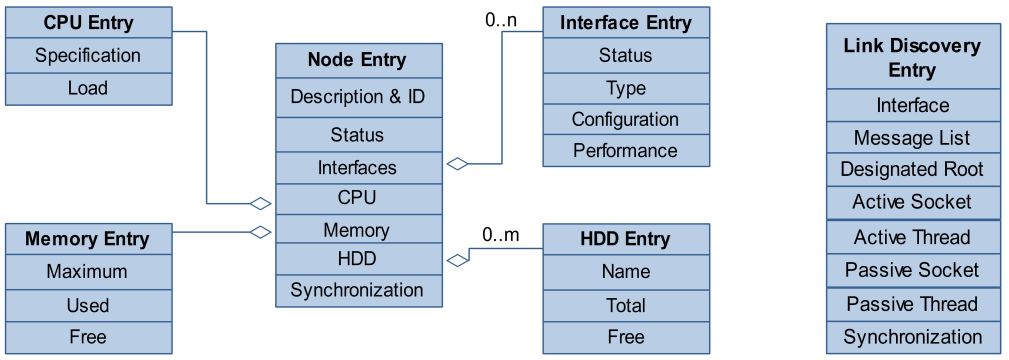


Figure 5.6: Connected Agents and Control Centers Entries.

5.2.3 Agent

Just like the Manager, the Agents also use linked lists to store their data. There are two main databases: The *Main Resource List* stores the information of all local resources in a linked list, while the *Neighbor List* stores information about the Agent's neighbors, also in a linked list. The neighbor database encompasses both physical and virtual neighbor information.



(a) Node Entry.

(b) Link Discovery Entry.

Figure 5.7: Node and Link Discovery Entries.

The access to each database is controlled by individual mutexes.

As can be seen in figure 5.7(a), each node entry contains information about the resource's CPU, RAM, HDDs and interfaces. There is also additional information about the resource, such as its ID and status. Synchronization variables deal with the concurrent access to the resource, since multiple threads may try to access it at the same time.

Auxiliary Databases

Because the discovery mechanism requires a pair of threads for sending and receiving per interface, a linked list exists containing control data structures. These control structures contain relevant thread control variables and a message list for inter-thread communication, as well as the required data for the discovery algorithm, such as the Designated Root ID, as can be seen on figure 5.7(b).

5.3 Type of Resource Identification

The NVSS as described in the State of the Art considered that the virtual and physical resources for the network nodes could be of 4 different types: router, server, switch and undefined. Although this was considered, physical resources would not be differentiated as their type was not identified by any means. This meant that all physical resources would be considered of the undefined type and allowed for any physical resource to host any type of virtual resource. The virtual nodes' type was also not registered after they were created in a physical node, therefore loosing the information submitted by the Control Center regarding the node's type. This allowed the NVSS to support a physical network where differentiation of resources was not needed and to map and enforce virtual networks with only virtual routers as nodes.

5.3.1 Server Identification

One of the new features specified for the NVSS is the support for creation and management of cloud resources, such as virtual servers. This requires the information submitted by the control center to the Manager regarding the node type to be registered when the node

is created, so that it is available when information about the virtual network needs to be collected and sent to the Manager. It also requires that physical nodes which host virtual routers are treated differently from physical nodes whose function is to host virtual servers, such as the case of data centers.

In order to do this, a new field for the type of the resource was added to the the XML file that registers the information about each virtual node in the physical node where they were created. The functions from the Agent module that wrote and read that information were also configured so that they would read and write that field together with the fields of the virtual nodes that were already being wrote and read at node creation and node discovery.

5.3.2 Type of Node Identification

In order to allow the physical resource type to be identified, a file was placed in the *xml_path* folder with a name that allowed for a simple and quick retrieval of this data. Besides providing information about the type of resources (i.e., if it is a router, server, switch or undefined), these files also provide information about the type of routers that the physical routers represent in the operator's network, which is an information that is considered when designing and mapping a VPN. This information is then used in a new attribute for the resources named TypeOfNode, which details the position of the node in the network, in the cases it is considered relevant.

5.4 VN & Cloud Configuration, Design and Mapping

When the user selects the Create New VNet command, the Control Center's View thread will create a new DisplayVNet object, and the related tab and drawing canvas. The following events are described in the next subsections.

5.4.1 Placing New Resources and Links

After selecting the Create New VNet command and specifying the name of the new VNet, the Control Center will show the tab associated to the new vnet, which is empty and where the user should place the virtual routers, servers and links to create a new virtual network. The drag-and-drop system allows the user to place new virtual servers or virtual routers as it wishes. For each router or server it places a new display object is created to place new resources and an object is created to manage the information of the resource that was just placed.

The user can also configure virtual links using the button of the coolbar associated with link creation. As the user presses this button an event is activated so that a dialog box appears requesting the origin and destination of the virtual link. The dialog box mechanism will check if the link already exists or if it has the same source and destination before proceeding. If so, a warning message appears and the user can select the link source and destination again or cancel the procedure. After selecting a valid source and destination the user can also set other parameters of the link such as bandwidth, latency and jitter. An object is created by the Control Center to be associated with each link that was placed and to contain its information.

5.4.2 Mapping

Upon committing the network, the Control Center will translate the information from a structure to a message and send it to the Manager.

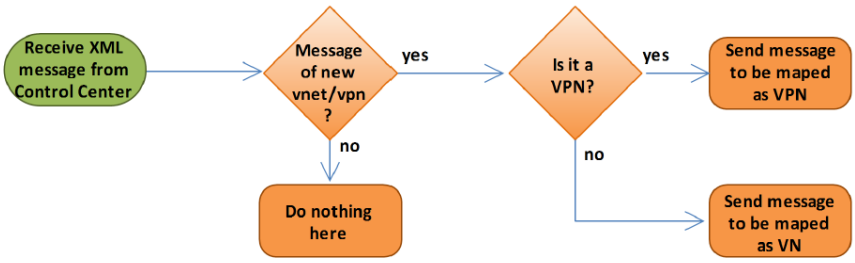


Figure 5.8: Flow diagram for when a new network is committed to the Manager

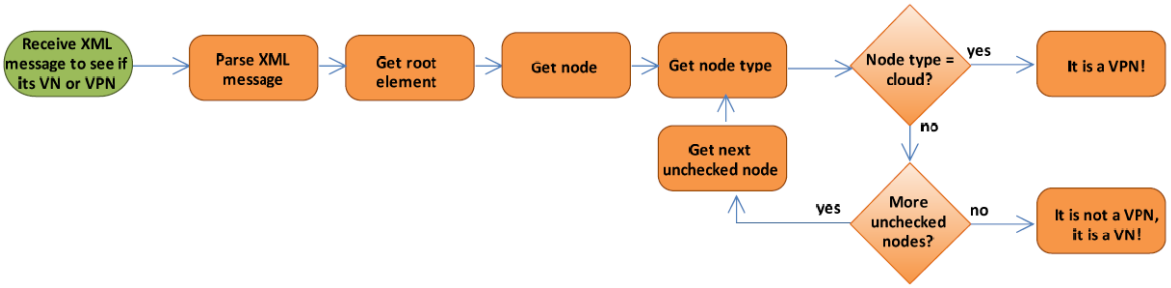


Figure 5.9: Flow diagram for the process of checking if the XML message received represents a new VN or VPN

The Manager will start by confirming if the message received is to create a new network. If it is, then it must verify if it corresponds to a VN or VPN. The resulting information will then be used to route the message to the correct mapping function, according to if it is a VN or VPN. This decision process is described in Figure 5.8.

If the XML message doesn't have any cloud node the Manager will consider it a VN, see Figure 5.9. As said before, when it verifies that the network is a VN, the Manager routes the XML message to the VN mapping function.

The first step of this function is to translate the XML message into a data structure. This translation process is long but simple, as it is merely the retrieval of values from the information fields of the XML message to the data structure, see Figure 5.10.

After this translation, the Manager will use the information provided by the Agents about the physical nodes to map the virtual nodes and links to the adequate physical nodes and links according to the algorithm proposed in the chapter Mapping Algorithms to map VN & Cloud, as described in Figure 5.11.

If this mapping is successful a message will be sent to the Control Center so that the user sees that the mapping procedure was successful. Otherwise a message will also be sent but announcing the unsuccessful mapping procedure. If the VN was mapped the Control Center will then send a message to the Agents with information about which virtual nodes and links should be set up to enforce the network.

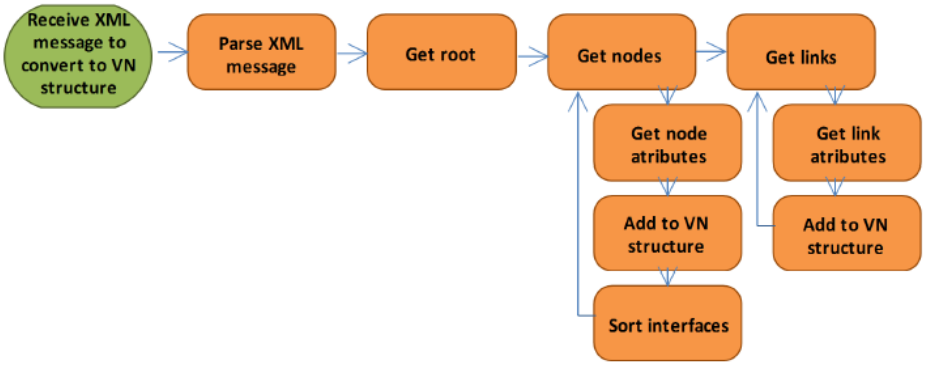


Figure 5.10: Flow diagram for the conversion of the XML vnet to structure

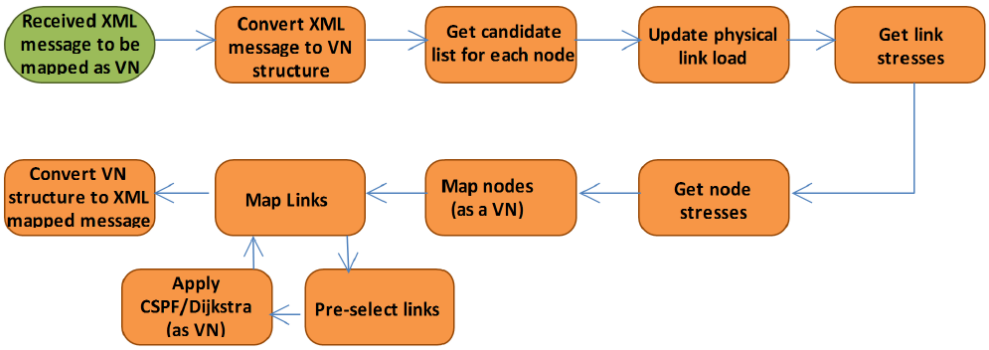


Figure 5.11: Flow diagram for the VN mapping process

5.4.3 Monitoring

As the Agents start creating virtual nodes and virtual links, the changes in the physical and virtual resources status will be detected by the monitoring thread, and the Agents will send information to the Manager about their current status. This information will then be sent to the Control Center so that the user is able to view what is happening with physical and virtual networks.

5.5 VPN & Cloud Configuration, Design and Mapping

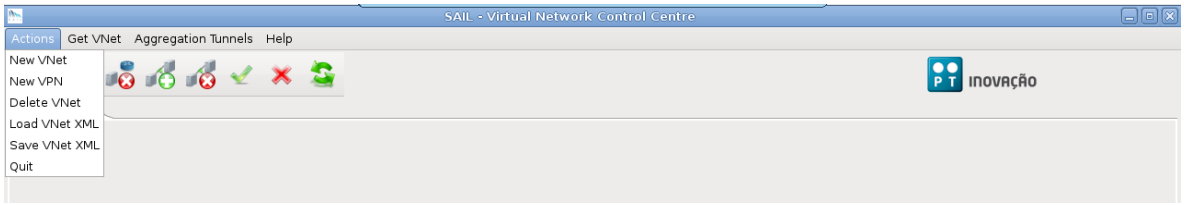


Figure 5.12: Options available from the dropdown Actions menu.

An option of "Create New VPN" was added to the Actions menu of the Control Center to

allow the users to configure VPNs, see Figure 5.12. When the user selects the Create New VPN command, the Control Center's View thread will create a new DisplayVNet object, and the related tab and drawing canvas. The canvas will not be empty as a new object representing the inner network of the VPN is created and displayed as a red router with "VPN" written on it, as in Figure 5.13.

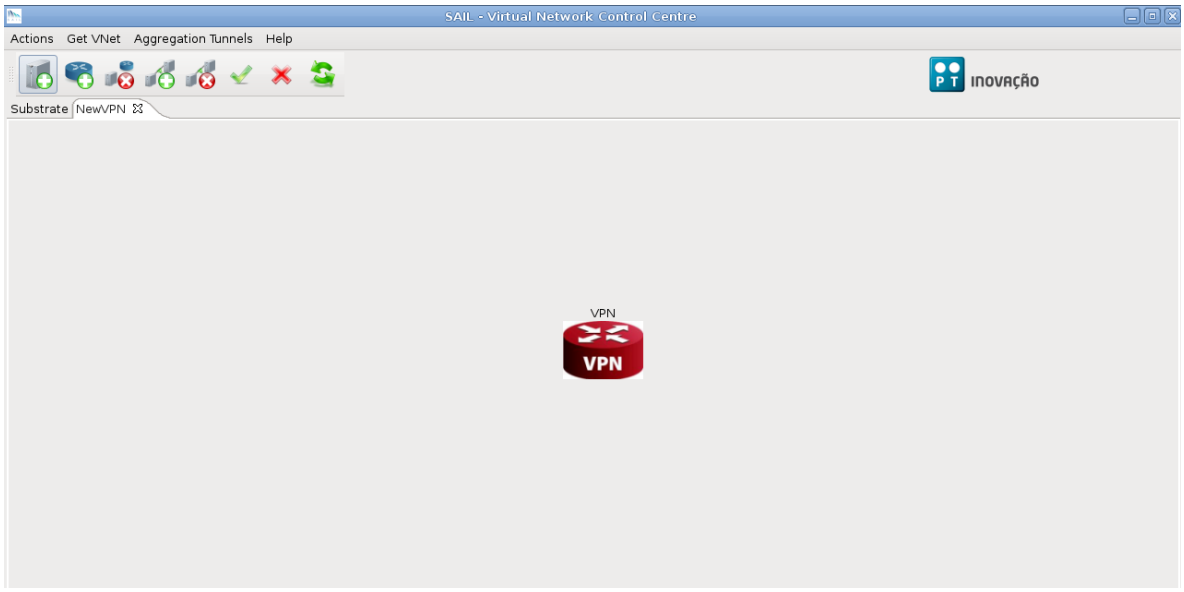


Figure 5.13: Initial state of the tab for a new VPN

This element has the same characteristics as a common virtual router except for the type which is named "cloud", but it's function is only of representation, which means that the characteristics of the element are irrelevant. Its' aim is to allow the representation of the configured VPN in a hose model, where traffic in and out of the network are the references and not the full topology. The following events are described in the next subsections.

5.5.1 Placing New Sites, Cloud Resources and Links

After selecting the option to Create New VPN the user will be able to place the sites that it wishes to connect to VPN, the Cloud resources it wishes to use and the links connecting both the sites and the cloud resources to the VPN network, represented by the red router already displayed on the tab of this VPN.

If the user wishes to place new sites it can do it by using the drag-and-drop system on the elements that represent virtual routers. In the VPN creation mode these elements represent the sites of the client which he wishes to connect. The user should configure their location after placing them by right-clicking on them, as the location will be the key data to decide where to map the virtual network extremes which will connect to those sites. As the user does this the View Thread will create new display objects so the placed objects will appear in the canvas and new display objects will be available to be placed. In parallel new elements containing the data regarding those resources are created and saved.

In order to place new cloud resources the user should do the same thing as it did with the sites, but now placing virtual servers. These elements are configurable by right-clicking

on them, which enables the user to set their HDD memory, RAM, number of cores, among others.

As each site or virtual server is placed, the link creation dialog box is automatically activated, so that the user can connect the site or virtual server to the VPN and select the amount of traffic going in and out the VPN from and to that point.

5.5.2 Design and Mapping of the corresponding VN

Upon committing the network, the Control Center will translate the information from a structure to a message and send it to the Manager.

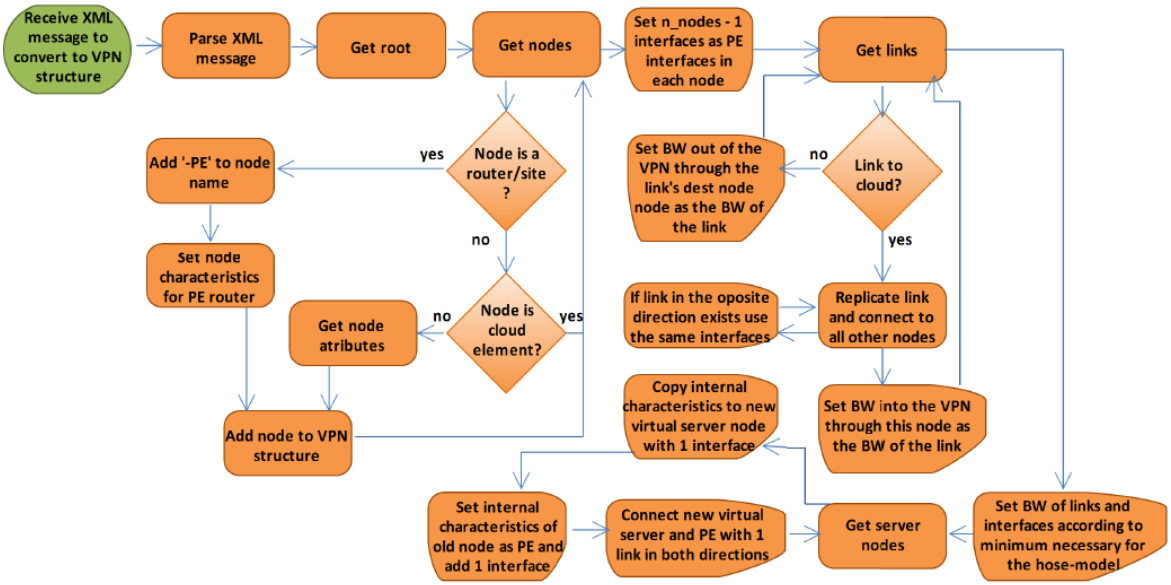


Figure 5.14: Flow diagram for the conversion of the XML VPN to structure

The Manager will then verify if it has received a VN or a VPN request, checking if there is any resource in the XML message of the cloud type, and then it will translate it to the appropriate structure. This is the same that was described for VNs in Figures 5.8 and 5.9.

If there is a cloud element in the XML message, the Manager will consider it a VPN and will convert it into a network where there is a Provider Edge router for each site and virtual server. Then, it will create a full mesh between the Provider Edge routers and a link between each virtual server and the corresponding Provider Edge router. This is Step 2 of Figure 5.15, which represents the steps from the configuration of the desired VPN-like network to the mapping of nodes and links. The translation process is represented in Figure 5.14. For each site, the corresponding Provider Edge routers will store information regarding the site location, but a virtual link will not be created between Provider Edge router and the corresponding site, since the site of the client is not controllable by the platform so that the virtual link can be built.

After this procedure the Manager will pre-map the virtual network that it built on the previous step.

This means that it will map the virtual network nodes as if this was a normal virtual network except for the following differences: Provider Edge virtual routers can only be

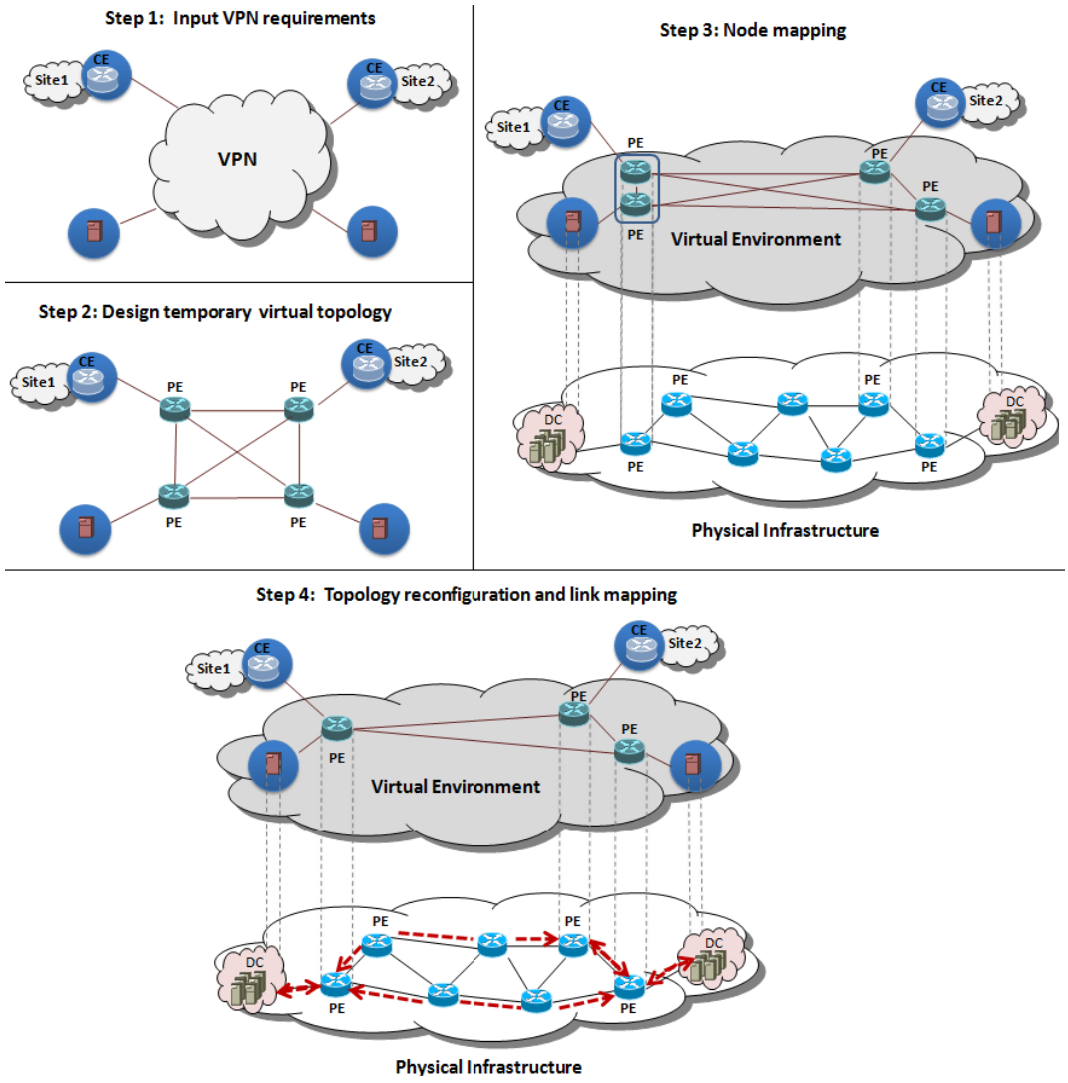


Figure 5.15: Steps from receiving the XML message from the visualizer until the mapping of both nodes and links in a VPN-like case.

mapped in Provider Edge physical routers; different Provider Edge routers can be mapped on the same physical Provider Edge router; the link cost will be 0 for virtual links where no physical link is involved, which is the case of the virtual links between Provider Edges hosted by the same physical Provider Edge router. This is represented in the third step of Figure 5.16. The larger mapping process, including the translation of the XML message is depicted in 5.15.

Using the information of this pre-mapping the Manager will remove virtual Provider Edge routers from the virtual network until there are no virtual Provider Edge routers sharing the same physical Provider Edge routers, removing also the attached links. This process is represented in Figure 5.17.

After doing this, the Manager will reset the full mesh between the remaining virtual Provider Edge Routers (now only one virtual Provider Edge Router per physical node) and

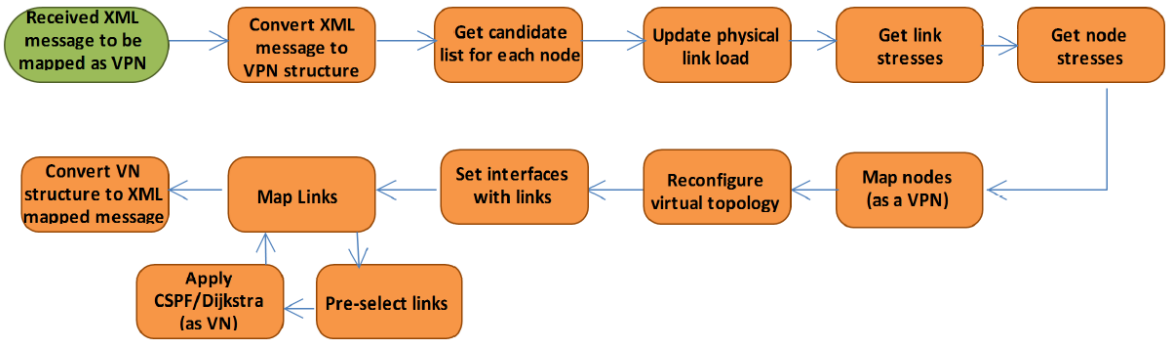


Figure 5.16: Flow diagram for the VPN mapping process

create links from them to the virtual servers whose virtual Provider Edge Routers were removed. Each virtual server is connected to the virtual Provider Edge Router that is hosted on the physical Provider Edge where the virtual server's Provider Edge Router was mapped. The bandwidth of all the links is reset to the appropriate values to comply with the hose-model.

After resetting the virtual network and having all the virtual Provider Edge routers and virtual servers mapped from the pre-map phase, the Manager will then proceed to map the virtual links connecting the virtual nodes. Both the virtual topology reconfiguration phase and the link mapping stage are depicted in Step 4 of Figure 5.15. After nodes and links are mapped the Manager will create a message with the mapping information and send it to the Agents so that they will create the virtual routers and servers and also the virtual links.

5.5.3 Monitoring

As this is enforced by the Agents, the changes in the resources status and topology are noticed by the monitoring thread of the Agents. The information about them is then sent to the Manager which will send it to the Control Centers so that the user is able to monitor the physical and virtual networks. At this stage, the user sees the virtual network and resources as it would see in a common VN: virtual routers and virtual servers connected through virtual links.

5.6 Conclusion

In this chapter the main databases, structures and classes of the three modules of the NVSS were described. It was also made a high-level description of the sequence of actions that enforce each new feature of the virtualization platform.

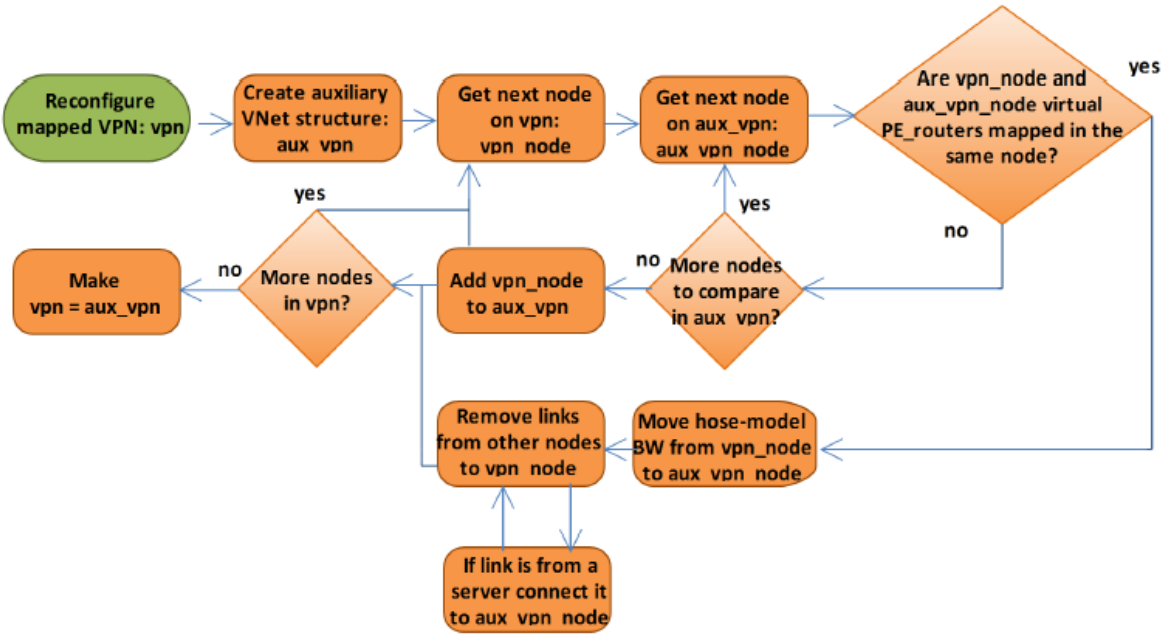


Figure 5.17: Flow diagram for the reconfiguration mechanism of VPNs.

Chapter 6

Experimental Tests & Results

6.1 Introduction

In this chapter we will present and analyze some experimental results related with the performance of the developed virtualization platform and with the algorithms proposed. We will also test the impacts of network virtualization on virtual network traffic.

We will start by describing the experimental testbed in 6.2. In 6.3 we will evaluate VN & cloud and VPN & cloud mapping decisions. This will be made my mapping consecutively 39 similar VNs and VPNs of 3 nodes (1 virtual server + 2 virtual routers) and analyzing the results in terms of node occupation and placement decisions.

In the next section, 6.4 we want to evaluate the dependence of mapping times from the number of nodes on the VNs and VPNs. Thus, we will map VNs and VPNs from 2 to 4 nodes (1 virtual server and the remaining virtual routers) without any preexisting virtual networks in the physical network and analyze the mapping times.

Finally, an experiment with different numbers of virtual networks and different values for traffic bandwidth will be presented and the practical impact on traffic shaping will be studied in 6.5.

The chapter ends with the discussion and overview of the main results in 6.6.

6.2 Testbed Description & General Assumptions

The testbed is composed of 6 physical nodes and is connected according to figure 6.1, obtained from the developed virtualization platform and modified to indicate the nodes roles.

The main specifications of the substrate nodes may be found in table 6.1.

6.3 Mapping decisions

Different proposals have been presented for virtual network mapping algorithms. In this section we will see how the algorithm we proposed in section 3.2 performs in an experimental testbed. The quality of the mapping process will be evaluated, in order to access the performance of the proposed algorithm in an experimental environment.

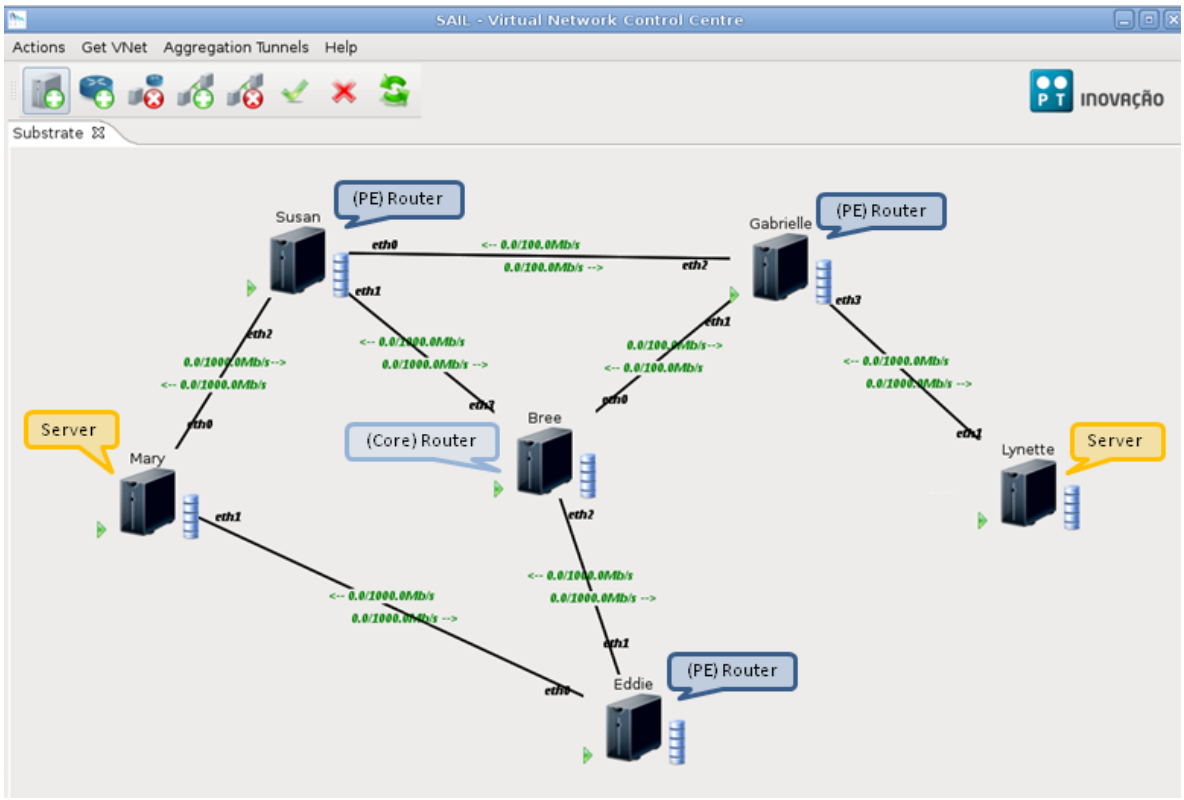
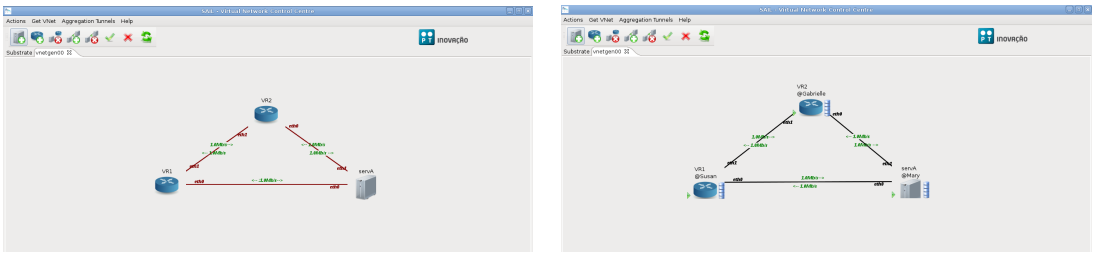


Figure 6.1: Testbed Network with indication of the machines roles.

Node	Susan	Lynette	Gabrielle	Bree	Eddie	Mary
CPU Model	Intel PentiumD 950	Intel PentiumD 950	Intel Core 2 Duo E6400	Intel Xeon E3110	Intel Xeon X3220	Intel Xeon X3330
CPU Freq.	3.40GHz	3.40GHz	2.13GHz	3.00 GHz	2.40GHz	2.66GHz
CPU Cores	2	2	2	2	4	4
CPU Threads	4	4	2	2	4	4
HDD Memory	89GB	40GB	145GB	195GB	303GB	277GB
RAM Amount	6GB	6GB	4GB	6GB	6GB	6GB
RAM Freq.	533MHz DDR2	667MHz DDR2	533MHz DDR2	667MHz DDR2	667MHz DDR2	667MHz DDR2

Table 6.1: Testbed specification.

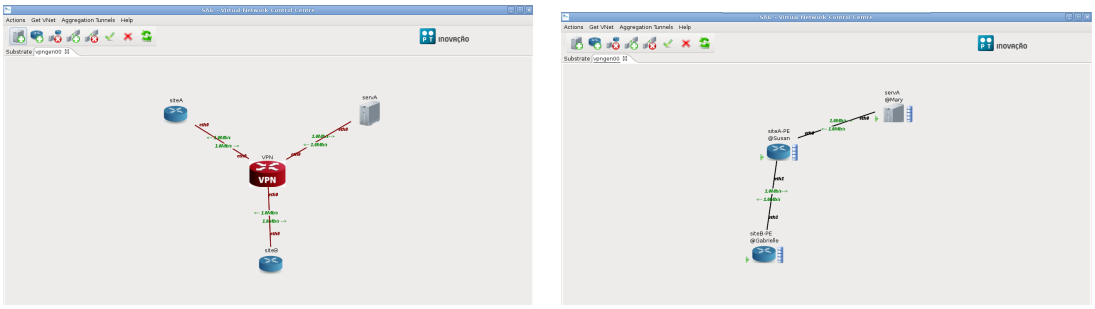


(a) VN to be mapped (b) Output example

Figure 6.2: Input and output case of the mapped VN

6.3.1 Methodology

For the first part of this experiment we have designed a standard virtual network to be mapped, which is presented in figure 6.2. Virtual node VR1 is restricted to be mapped either in Susan or in Eddie, by using geographical restrictions, while VR2 is restricted to be mapped on Gabrielle. Virtual server ServA can be mapped either in Mary as in Lynette. We erased all pre-existing virtual networks and mapped 39 of this virtual networks in sequence, and repeated this process 3 times. Values of node occupation as a function of the number of pre-existing virtual networks were registered and analyzed.



(a) VPN to be mapped (b) Output example

Figure 6.3: Input and output case of the mapped VPN

For the second part we have designed a VPN request with 2 sites and 1 virtual server. SiteA location is such that its PE Router must be either in Susan or Eddie, while for SiteB its PE Router must be in Gabrielle. Virtual server ServA can be mapped either in Mary or in Lynette. The VPN input figure is represented in Figure 6.3.

As before, values for node occupation as a function of the number of pre-existing virtual networks were registered and analyzed. The graphs present average occupation values and 90% confidence intervals.

6.3.2 Tests & Results

VN & Cloud

In Figure 6.4 we can see the occupation in the server machines. In each mapped VN there is a virtual server which can be mapped either in Mary or in Lynette. We can see

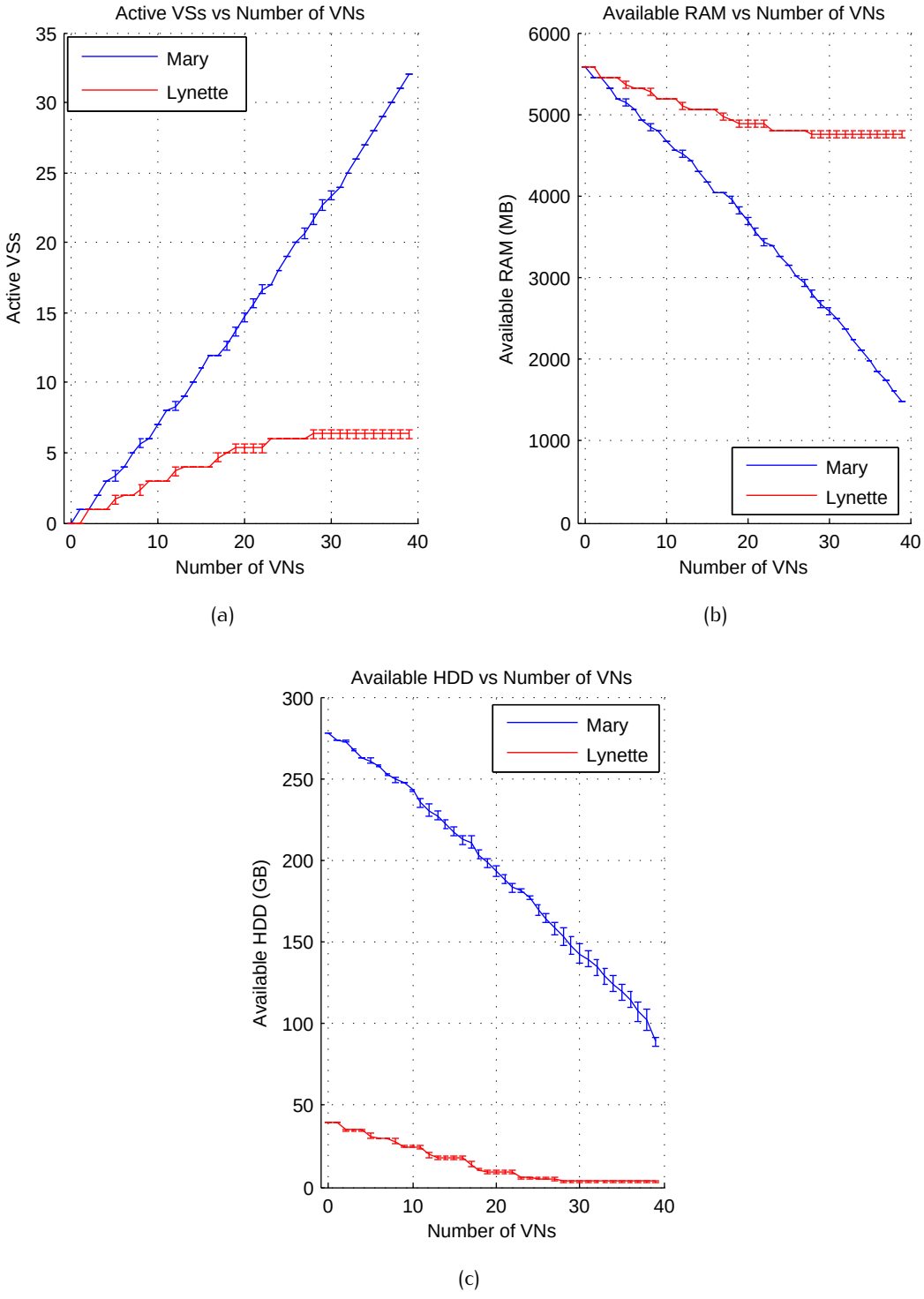
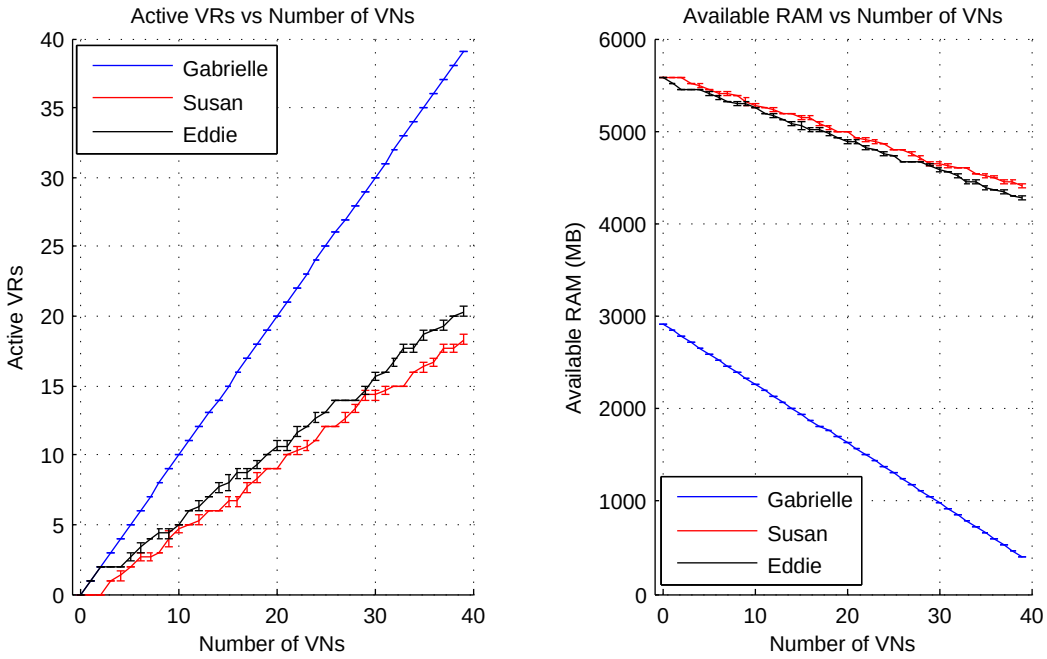


Figure 6.4: VN& Cloud Mapping: Occupation of Physical Servers

that Lynette has a lot more free HDD memory than Mary, thus expectantly making it more prone to host the virtual server of each mapped VN. This is confirmed by the experimental

results, where we can see that the virtual servers tend to be hosted by Mary, while Lynette only gets a smaller portion of them. As Lynette's free HDD gets smaller, the frequency with which it hosts new virtual servers is also reduced.



(a) (b)

Figure 6.5: VN& Cloud Mapping: Occupation of Physical Routers

Meanwhile, Susan and Eddie dispute one of the virtual routers. In Figure 6.5, we can see that both get a similar number of virtual routers, with a regular difference in occupation, where Eddie is slightly more occupied than Susan. This is to be expected, as Eddie has the double number of cores of Susan, thus making Eddie more prone to accept more virtual resources. It should be noted that Susan has a higher CPU frequency than Eddie, but it is not even near double the CPU frequency of Eddie.

VPN & Cloud

In Figure 6.6 we can see the occupation in the server machines. As in the previous case of the VNs, we can see that Mary hosts more virtual servers than Lynette, which is expected since Mary has more free resources than Lynette, especially HDD memory.

But when we get to look at Susan and Eddie in Figure 6.7, which dispute one of the virtual PE routers of each VPN, it is very curious to see that now it is Susan which is more occupied than Eddie, even though Eddie has more resources than Susan. Looking more carefully into the testbed topology and the virtual topology of VPNs it is possible to explain why this happens. While in the VN case the virtual server had to connect to 2 virtual PE routers, in this situation the virtual server only has to connect to one virtual PE router. This way, when mapping the virtual nodes in Gabrielle and Susan and the virtual server Mary, we use a small set of links, since Susan has direct connections both to Mary as to Gabrielle.

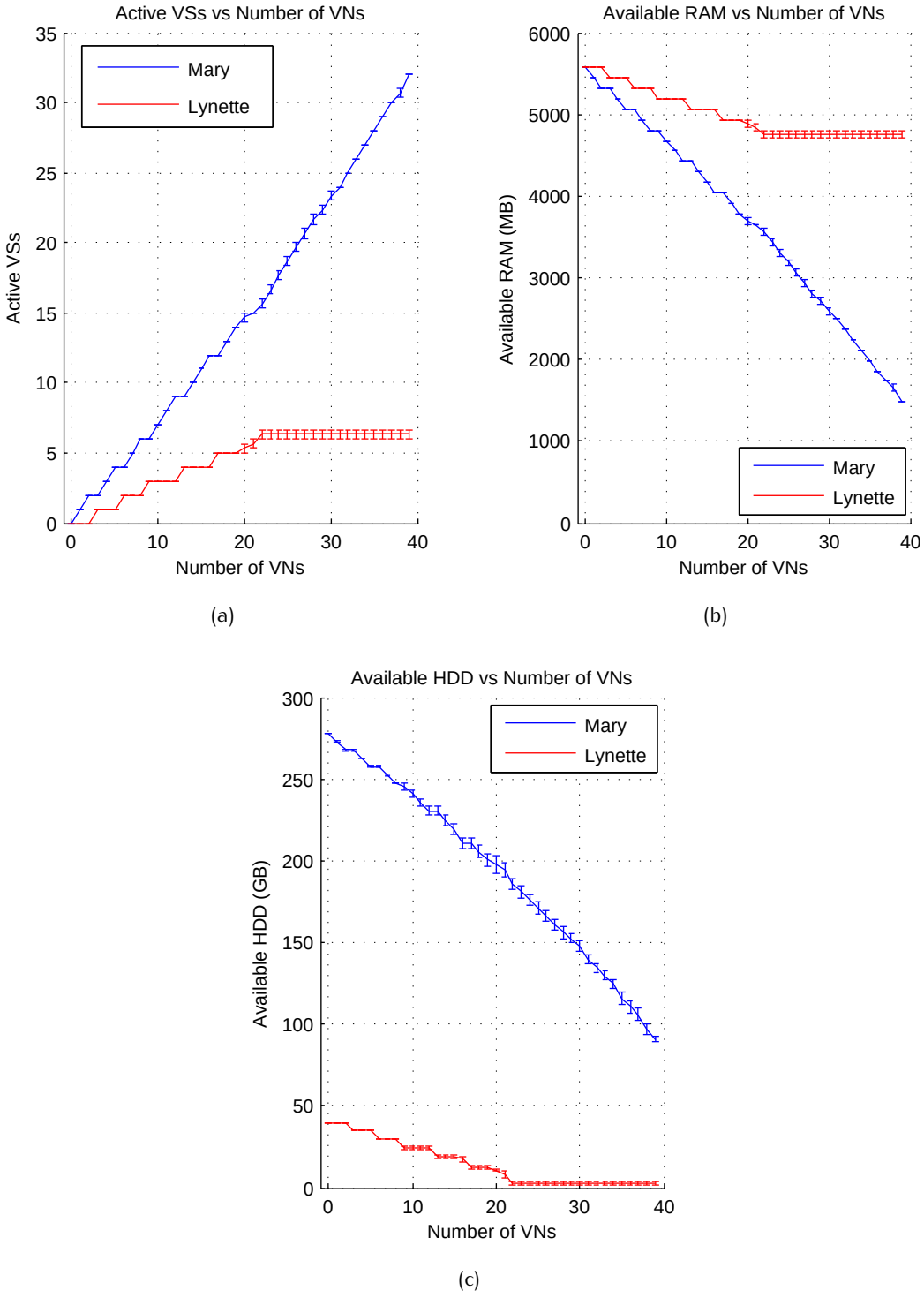
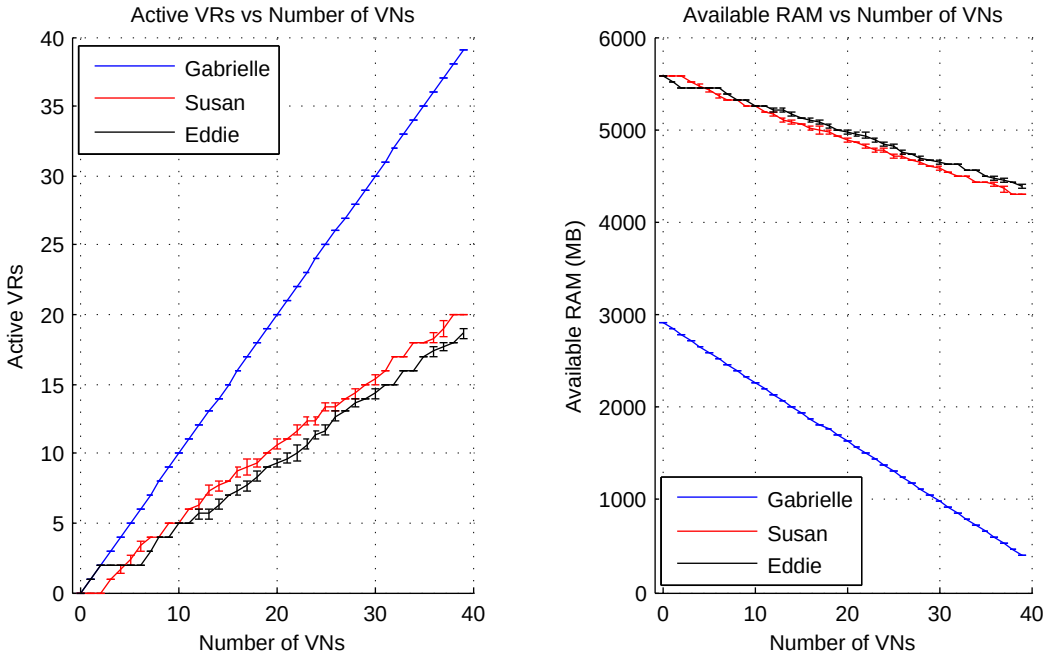


Figure 6.6: VPN& Cloud Mapping: Occupation of Physical Servers

This smaller link cost will compensate a higher node stress, and thus make Susan more prone to host virtual servers. In the case of the VNs, mapping the nodes in Susan and Gabrielle



(a) (b)

Figure 6.7: VPN& Cloud Mapping: Occupation of Physical Routers

and the virtual server in Mary does not reduce the link cost enough to compensate higher node stress, since Mary will have to have a virtual link to Gabrielle, which will be mapped via Susan (since it is 'cheaper' than via Eddie), thus occupying more of these physical links which are considered to calculate Susan's link cost.

6.4 Mapping times

Using simulation we can test the decisions of the mapping algorithms, but parameters such as mapping time can only be obtained with some accuracy in an experimental testbed. This section's objective is to evaluate the mapping times for the proposed algorithm implemented in the platform.

6.4.1 Methodology

In this second part we will map different VNs and VPNs onto the physical network with no previous virtual networks embedded, changing the number of virtual routing and servers nodes and registering the respective impact on computing times. The input VNs are depicted in Figure 6.8, while input VPNs are depicted in Figure 6.9. Each of these was mapped 5 times in a physical network with no virtual networks previously embedded. Using geographical restrictions, we assured that for each site of the VPNs a virtual PE router was being created, so that the number of nodes being mapped and created were the same and corresponded to the number on the X axis of the graphics.

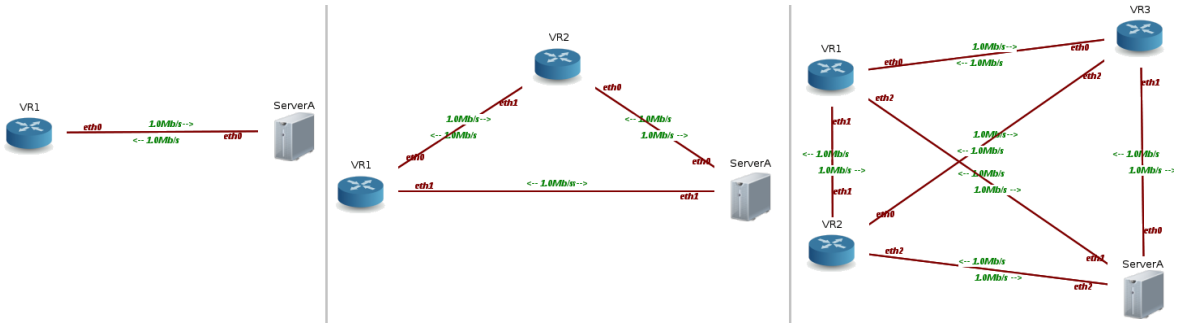


Figure 6.8: Set of VNs to be mapped.

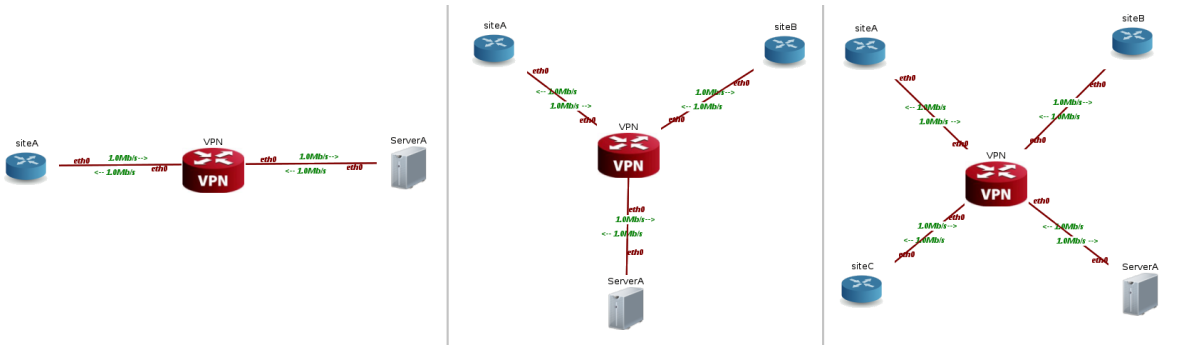


Figure 6.9: Set of VPNs to be mapped.

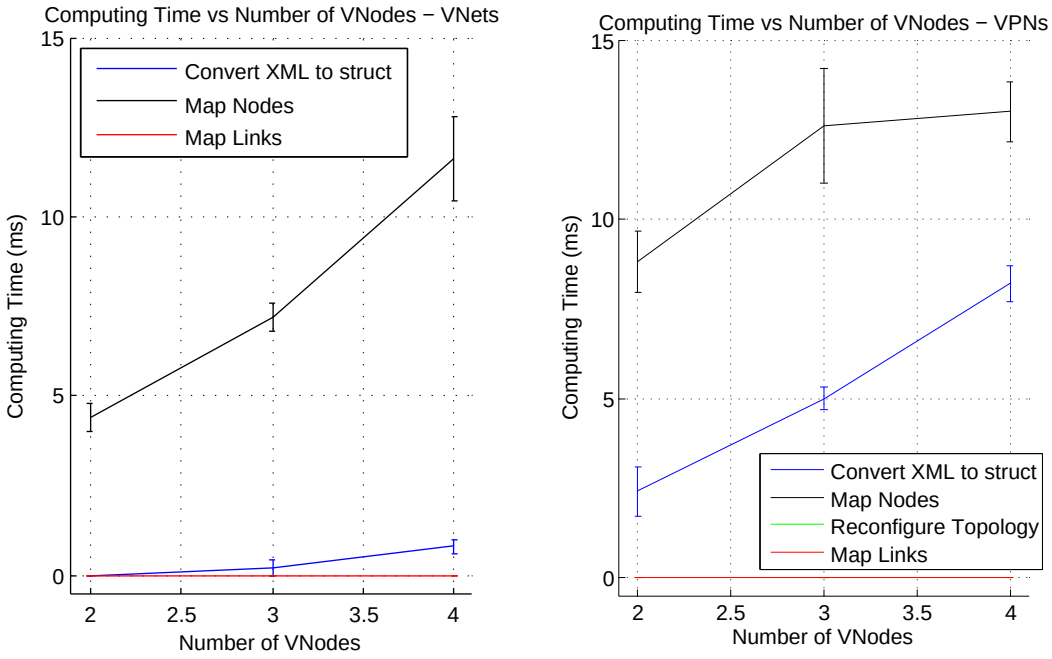
Mapping times were registered and analyzed. The graphs represent average mapping times with 90% confidence intervals.

6.4.2 Tests & Results

In Figure 6.10 we can see the computing times for the different steps from receiving the XML message of a VN or VPN to the moment that network is mapped. The sequence is:

- 1. Conversion from XML to structure
- 2. Mapping the nodes
- 3. Reconfiguring the virtual topology (only if it is a VPN)
- 4. Mapping the links

Figure 6.10(a) shows the results for the mapping times of VNs. We can see that the time to convert from the XML message to a structure increases with the number of nodes, but is still a small value when compared to the time to map the nodes. While XML conversion times for VNs go from 0 to 1 ms when we go from 2 to 4 node VNs, times for node mapping go from around 4.5ms to 11.5ms. Node mapping times seem to increase in an almost linear fashion with the increase in the number of virtual nodes. Link mapping times are always null on this graph because our time unit while measuring was milliseconds. This means that link mapping is very fast and takes less than 1 millisecond, which is a lot less than the time required for some of the other steps.



(a) Computing times for the mapping of VN & Cloud with 2, 3 and 4 virtual nodes

(b) Computing times for the mapping of VPN & Cloud with 2, 3 and 4 elements (sites and virtual servers)

Figure 6.10: VPN& Cloud Mapping: Computing times

As for the mapping of VPNs in Figure 6.10(b), we can see an extra time stage, which is the time used to reconfigure the virtual topology after the nodes have been mapped. This line, which is green, is not visible because it is also at a 0ms value in all of the cases, and thus is below the red line. We can see that conversion from XML to structure takes between 3 to 14ms in VPNs, which is a much longer time than in VNs, where the maximum time for the same number of nodes is 1ms. This is a result of the fact that in order to create the topology of a VPN the manager does not only translate an XML file into a structure, it also creates and sets most of the virtual network, as when it creates virtual provider edge routers and links in order to establish a full-mesh between these routers. As for node mapping times, they are higher in the VPN case, ranging from 9 to 13ms. This might be explained by the fact that the node mapping process in VPNs is made using the temporary virtual topology, which includes 1 PE router per site or server + 1 server element per virtual server. Link mapping on both cases is always close to 0. This might be due to the efficiency of the Dijkstra algorithm and the small number of nodes both in the virtual network as in the testbed. Times for topology reconfiguration in the VPNs are also very small, which might be explained by the simple nature of this action, which mainly consist in removing some virtual nodes and links.

It should be noted that, since the conversion of the XML messages to structure is not part of the mapping algorithm, and times for topology reconfiguration and link mapping are always smaller than the smallest time unit considered, the mapping time of the networks has the same value as the time of the "Map Nodes" step.

We can see that node mapping times, which coincide with global mapping times, are higher for VPNs than for VNs, which is expectable since there are more resources to be mapped in VPNs, even if some of them are removed later when the virtual topology is reconfigured.

6.5 Virtual Traffic Analysis

Ideally virtual networks should behave just as independent physical networks, but mostly we get an approximation to this behavior. In this section we want to observe how the changes in the number of virtual routers, virtual links and amount of traffic coexisting in the same physical network influence the performance of the virtual networks. Thus, we will register and analyze parameters of first and second order of packet delay times, such as average packet delay, packet delay variation and packet delay variance.

6.5.1 Methodology

In order to do this we designed a simple experiment to have multiple virtual networks with the same topologies coexisting in the same physical network. This is depicted in Figure 6.11. Three physical machines are used: at the start and end of the network we generate and receive flows of traffic in different interfaces, so as to simulate different (virtual) networks. In the center of the network a machine runs one Virtual Router for each traffic flow involved, the same is to say for each Virtual Network in use.

In order to build this system the three machines were initially connected through 2 ethernet connections, 1 from Mary to Eddie and another from Eddie to Bree. Using the NVSS platform 3 virtual networks were created with the same topology as the physical topology of this experiment. This way, 3 virtual routers were created in Mary, 3 in Eddie and 3 in Bree. In order to connect each set, 3 virtual links were created from the 3 virtual routers in Mary to the 3 virtual routers in Eddie, the same being repeated from Eddie to Bree.

After this the virtual routers running in Mary and Bree were destroyed, together with the virtual interfaces. Three Iperf[58] sessions were used in Mary so as to generate 3 flows of traffic, each sending the packets through a previously attributed ethernet interface. Similarly at Bree, 3 Iperf sessions were used in order to receive the incoming flow of traffic, each one capturing packets from a specific ethernet interface. Two switches supporting VLAN tagging were connected between the Mary and Eddie, and Eddie and Bree, so as to apply a certain VLAN tag in each packet to associate each flow of traffic with a virtual link and virtual network. This way, when the packets arrive at Eddie, the machine routes each packet to the appropriate virtual interface according to the VLAN tags and existing bridges, so that the packets get processed by the corresponding virtual routers.

Two hubs are used in order to monitor the packets going in and out the virtual routers, in order to measure the delay times when entering and exiting the virtual routers. The Wireshark[23] application was used to measure these times. Wireshark places a computer time stamp in every packet collected. Using the packets' data field, a match was made between packets captured going in and out of Eddie, thus allowing for time comparisons using the time stamps.

In this experiment we start with 1 virtual network and go on until we have 4 virtual

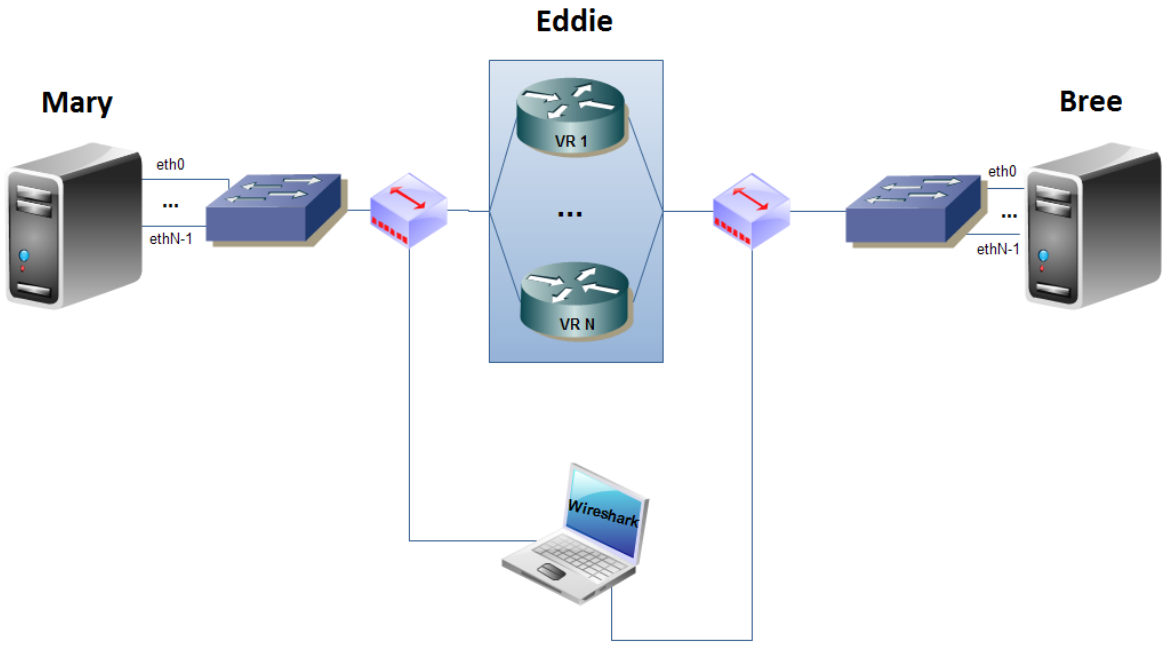


Figure 6.11: Experimental apparatus

networks coexisting in the same physical network. Two modes are used for the bandwidth of the data flows: in the first one we keep the total traffic at 3Mbps (thus each flow uses this bandwidth divided by the number of flows) while in the second one we set 1Mbps for each flow (thus increasing the total traffic on the physical network proportionally to the increase on virtual networks).

The traffic generated was made of UDP packets of 1300 bytes sent at a constant bit rate. Eddie uses Xen[64] as the virtualization software, release 2.6.21.7-5.fc8xen, with the credit scheduler option selected. 5 runs were made for each set of values, with 30 second samples of traffic being captured and analyzed. Graphs show the average values of the 5 runs with confidence intervals for a 90% probability.

6.5.2 Tests & Results

In Figure 6.12 we can see the average packet delay times evolution as we change the number of virtual routers and virtual links coexisting in the same physical machines and physical links. For both modes, these values remain constant and have little difference between them: we can see an almost horizontal line for the average values and the confidence intervals are small. This means that average packet delays remain constant when the amount of traffic on the virtual networks changes and also when the number of networks changes. It should be noted that these observances are only valid for this set of values for traffic and number of Virtual Routers (VRs). In order to be able to generalize this conclusion on should experiment with more values of traffic and with higher numbers of coexisting virtual networks.

In Figures 6.13 and 6.14 we can see two indicators for the second order statistics of packet delay: packet delay variance and jitter. Packet delay variance is, as the name says,

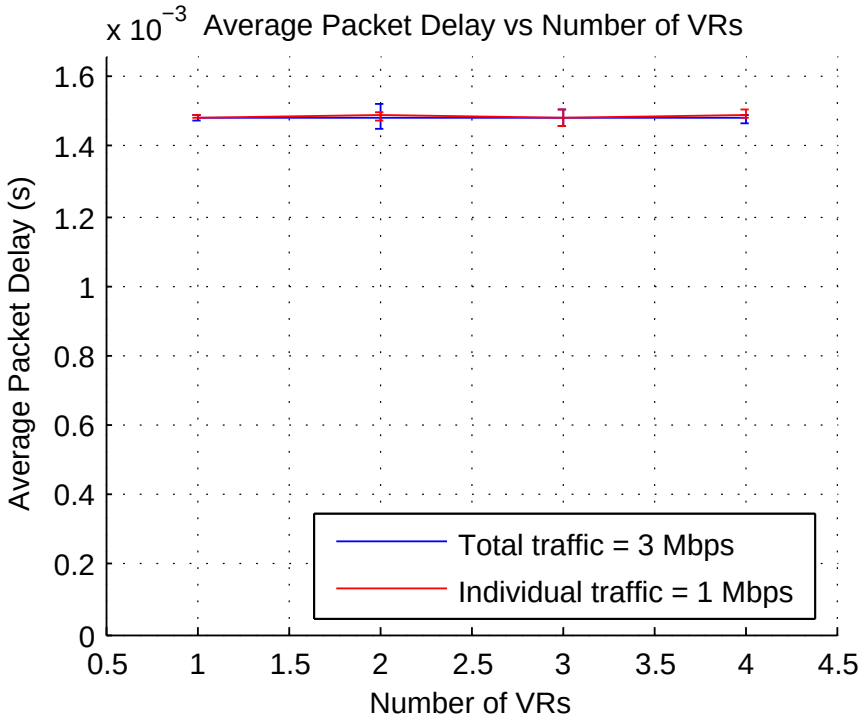


Figure 6.12: Average packet delay for different numbers of VRs for both modes

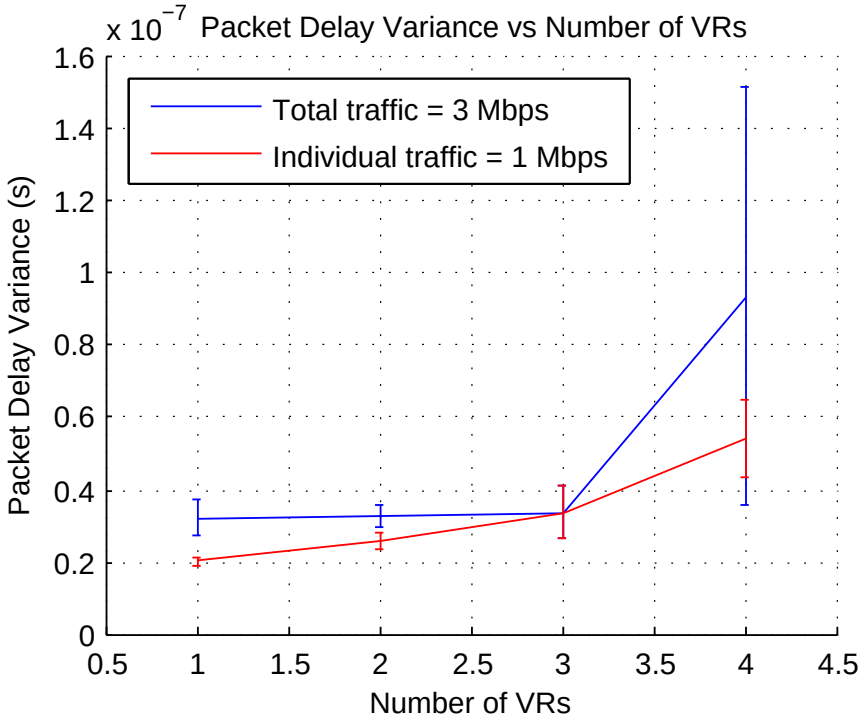


Figure 6.13: Packet delay variance for different numbers of VRs for both modes

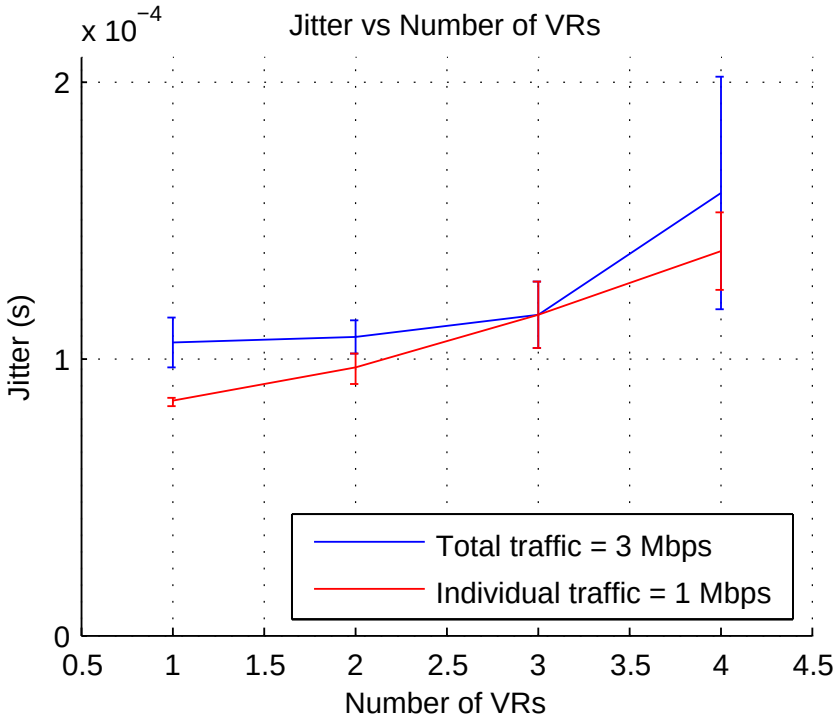


Figure 6.14: Jitter for different numbers of VRs for both modes

the variance of the packet delay times in the different runs. Jitter is calculated here as the average absolute difference from each packet delay time to the average packet delay time. When maintaining the same amount of total traffic in the physical network, we can see in both indicators increasing as the number of virtual routers and links increases. It is also possible to see that the confidence intervals increase with the increase in the jitter and variance values, which means that average values get higher and there is more "instability" in these values. As for the other mode, when we proportionally increase the amount of total traffic in the physical network, we can see an approximately linear variation of packet delay variance and jitter with the number of virtual routers and links. In this mode, confidence intervals increase with the increase in the jitter and variance values (which increase with the number of virtual routers and links).

If we compare the results for both modes, we can see that the variance of packet delay times is usually larger for the mode where the total traffic is set at 3Mbps. This should be expectable for 1 and 2 VRs when having 3Mbps for 1 or 2 virtual routers and networks means having 3 or 1.5 Mbps in each VR, which is more than 1 Mbps for each VR in the other mode. At 3 VRs the modes and delay times are the same, since having 3 Mbps through 3 VRs means that we have 1Mbps through each VR. For 4 VRs we would expect to have smaller variance of delay times for the 3Mbps total traffic mode, since this would mean less traffic per VR than in the 1Mbps per VR mode, an idea that is not confirmed by the data. Since we have a very large confidence interval in the mode of total traffic at 3Mbps for 4 VRs, which is much larger than all other confidence intervals, we might suppose that the set of values used might be not be representative of the usual values for this case due to its high degree of uncertainty. In order to verify this situation, more data should be collected

and analyzed.

From the data of this experiment, we can see that average packet delay times remain the same, even with changes on the amount of traffic in the physical network, in the virtual networks and virtual routers, and in the number of virtual networks. However, as the number of virtual routers and link increases, the packet delay times get more different from each other (which is indicated by a higher variance among these values), thus diminishing the quality of experience for communication services in real-time, such as voice or video calls.

6.6 Conclusions

In this chapter we have seen the performance results of the different features added to the NVSS platform. We analyzed mapping decisions for VNs and VPNs in 6.3 and saw how machines with more resources or with better connections for other physical nodes were able to host more resources. It was possible to see how network and node stress interplayed and influenced mapping decisions and how physical machines with a critical level of free resources tended not to be used. These observations confirm that the NVSS is making the adequate mapping decisions and the mapping algorithm is performing as it was designed to.

When we observed the dependence of mapping times from the number of network elements on both VNs and VPN in section 6.4, we could see that the time for converting XML messages to structures and to map nodes significantly increase when the number of virtual nodes grows. Times for VPN node mapping are significantly higher than for VN node mapping, which is expectable since the VPN node mapping is done using the temporary virtual topology, which includes a larger number of nodes than the used in the VN mapping. It was also verifiable that XML to structure translations times were much larger for VPNs than for VNs, which is due to the increased complexity of this operation in VPNs, since the Manager uses this stage to obtain a temporary virtual topology for the VPN, which does not happen in the VN case. Interestingly and perhaps unexpectedly, it was observed that computing times for link mapping and virtual topology reconfiguration (in the case of VPNs) are a lot smaller than other computing times (smaller than 1 millisecond). In the case of the link mapping, this is probably justified by the efficiency of the Dijkstra algorithm and the small number of nodes involved. As for the virtual topology reconfiguration, it might be speedily executed because most of the processing is just removing nodes and links, which is a simple action.

Results on the effects of virtualization on virtual traffic characteristics show that, for the set of values used in the experiment, average packet delay times remain the same even when the number of coexisting VNs and traffic values increase. In spite of this, variance of packet delay times seems to increase with the number of coexisting VNs and with the increase on traffic values. This reveals that coexisting VNs affect each other, which ideally should not happen. Future research on virtualization technologies should tackle this aspect, especially since packet delay variation is a critical factor for real-time communication services.

Chapter 7

Conclusions

7.1 Final Conclusion

This work had three main objectives: to develop an optimized algorithm for integrated cloud and network mapping, to enable the NVSS to support cloud resources more than just network resources and to allow the virtual networks to be configured and designed as VNs or VPNs.

In order to put everything in perspective, the State of the Art chapter provided some insight into the current deployment and R&D context regarding virtual network and cloud mapping, provision of cloud resources and VPNs, and existing network virtualization platforms and initiatives. The CC products of relevant operators such as Amazon, Orange and Portugal Telecom were presented. Virtual network mapping algorithms were also presented just as the NVSS, both representing the starting points for the algorithm improvement and creation of advanced mechanisms for the NVSS platform.

In chapter 3, Mapping Algorithms, several improvements were proposed to the algorithm described in [41], in order to support link differentiation, cloud resources and to improve the quality of the mapping solutions. In order to do this a discrete event simulator was built, and different algorithms were compared. In order to take into consideration the node placement options affecting nodes in distant places of the virtual network, a feature was added to the algorithm so that node candidates could be removed for not offering viable physical connections for certain virtual links. A back-tracking mechanism was also added to correct wrong placement choices. This revealed to improve the amount of virtual networks and virtual resources embeddable in the substrate network, while having a mapping time increase as trade off. Several other formulas concerning the calculation of node and link stress were also considered, and it was concluded that some node stress parameters could be omitted, some proposed node formulas were significantly better than the one in use, link stress formulas had similar performances. We were also able to see that, as the number of nodes of the virtual network increases, the likelihood that the network is embedded decreases while the mapping time increases.

The target advanced mechanisms and use cases to be added to the virtualization platform were specified on chapter 4. A description of each new feature was provided together use cases schemes. An overall description of the aimed NVSS was also provided.

In chapter 5, Software Implementation, the programming solutions for the mechanisms designed and proposed in chapter 4 were provided. We could see the databases, classes

and structures in use, the different modules organization and functions, and the implementation of the new functionalities. The main new functions added to the NVSS platform were described using schematic diagrams, in order to provide a simple high-level explanation of the implementation.

Chapter 6, Experimental Results, provided more insight about the experimental output of the new features on an experimental testbed. Results regarding mapping decisions and computing times of the mechanisms for VN & Cloud and VPN &Cloud processing and mapping were presented and discussed. We could see that nodes with more resources tended to host more virtual nodes, as desired, confirming the functionality of the mapping algorithm and its successful implementation in the NVSS. Mapping times analysis showed a clear increase in mapping times with the increase in the number of virtual nodes. It was also possible to see that computing times for VPNs are significantly higher for VPNs than for VNs. This is expectable as these networks include more virtual nodes to map and the step of XML conversion includes the design of the temporary virtual topology. Surprisingly, times for link mapping and topology reconfiguration in VPNs were always smaller than 1ms, which is significantly smaller than the computing times for other steps. This is a good indication, especially for VPNs, as it shows that the reconfiguration stage does not introduce a significant time over-head.

Besides testing these features we grabbed the opportunity and also made tests regarding the effect of virtualization on traffic characteristics. It was possible to see the effects of virtualization on virtual network traffic. As the number of coexisting virtual networks or bandwidth in use increases, so does the packet delay variation. In spite of this, average packet delay times remain constant even when the number of VNs or amount of traffic is changed.

In the end, we can see through the simulation results that the algorithms for virtual network were improved in three ways: the quality of the mapping was improved allowing for more networks and virtual resources to be mapped, the algorithm was extended so as to consider also cloud resources such as virtual servers, support for link differentiation criteria was also included by introducing the interdependency mapping feature that allowed for networks with differentiated links to be better mapped than before.

The NVSS was also improved as now it supports mechanisms for cloud resource configuration, mapping and enforcement integrated with virtual networks, and mechanisms to create virtual networks by specifying the hose model of the network.

Our knowledge regarding the effects of virtualization has also increased, as we were able to observe and quantify the increase in packet delay variation with the number and bandwidth of the coexisting virtual networks.

7.2 Future Work

There are several topics that can be further explored in the future related with this work.

On the one hand, mapping algorithms could be enhanced and a distributed approach could be explored. This would allow the network itself to perform the mapping, which would provide for a distributed mechanism that might possibly take advantage of the physical node's parallel computation to reduce mapping times.

The mapping mechanism developed and implemented on the NVSS considers that a virtual link uses the same bandwidth in both its directions. This might not always be true,

in the sense that upload and download links might have different bandwidth requirements. So, it should be developed and evaluated an efficient algorithm that allows for optimized node and link mapping in virtual networks with virtual links free to have different bandwidth requirements in each direction.

As for the NVSS several advanced mechanisms could be added. For instance, reconfiguration. Automatic reconfiguration, in the sense of remapping virtual nodes and links which are already embedded and running, could be studied and implemented so as to reduce the use of the physical resources in order to save energy and reduce costs.

On the other hand, it would probably be very useful to have the NVSS capable of adding nodes and links to virtual networks that are already embedded and running.

And one should not forget scalability. We can see in large-scale physical networks that different flows and tunnels are grouped and processed as a single flow in core routers, in order to achieve scalability in terms of routing. The same situation should be considered when creating several virtual networks over one large-scale physical network. Since having a routing entry for each virtual link in core routers is not scalable, the appropriate mechanisms should be devised so as to guarantee the scalability of the routing process. The NVSS platform could be used to aggregate links using, for example, MPLS as the supporting technology.

The tests for virtual traffic analysis could also be more explored, by embedding more virtual networks and trying different scenarios. This would allow for clearer observations and analysis on the effect of virtualization on traffic, its limitations, challenges and possibilities of improvement.

Bibliography

[1] 4WARD: *4ward*. http://www.4ward-project.eu/.

[2] AKARI: *Akari*. http://akari-project.nict.go.jp/.

[3] al., A. Bav ier et: *In vini ver i tas : Real i s t i c and cont r o l l e d n e t w o r k e xpe r i m e n t a t i o n*. In *P r o c . A C M S I G - COMM*, pages 3–14, 2006.

[4] al., J. E. van der Merwe et: *The tempest: A practical framework for network programmability*. In *IEEE Network, vol. 12, no. 3*, pages 20–28, 1998.

[5] al., M. Boucadair et: *A framework for end-to-end service differentiation: Network planes and parallel internets*. In *IEEE Commun. Mag., vol. 45, no. 9*, pages 134–43, 2007.

[6] al., M. Kounavis et: *The genesis kernel: A programming s y s t em f o r spawn i n g ne two r k a r c h i t e c t u r e s*. In *I E E E JSAC, vol. 19, no. 3*, pages 511–526, 2001.

[7] al., P. Ruth et: *Virtual distributed environments in a shared infrastructure*. In *Computer, vol. 38, no. 5*, pages 63–69, 2005.

[8] al., W. Ng et: *Miblets: A practical approach to virtual network management*. In *Proc. 6th IFIP/IEEE Int'l. Symp. Integrated Net. Mgmt.*, pages 201–215, 1999.

[9] Amazon.com: *Introducing Amazon Virtual Private Cloud (VPC)*, August 2009. http://aws.typepad.com/aws/2009/08/introducing-amazon-virtual-private-cloud-vpc.html.

[10] Amazon.com: *Media Kits: Overview*, February 2011. http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-mediaKit.

[11] Amazon.com: *What is AWS?*, June 2011. http://aws.amazon.com/what-is-aws/.

[12] Andersen, David G.: *Theoretical approaches to node assignment*. Unpublished Manuscript, December 2002.

[13] ARISTA, January 2011. http://www.aristanetworks.com/.

[14] Bouyoucef, K., I. Limam-Bedhiaf, and O. Cherkaoui: *Optimal allocation approach of virtual servers in cloud computing*. In *Next Generation Internet (NGI), 2010 6th EURO-NF Conference on*, pages 1–6, Junho.

[15] Carr, Nicholas: *The Big Switch*. http://www.nicholasgcarr.com/bigswitch/.

[16] Chowdhury, N., Muntasir Rahman, and Raouf Boutaba: *Virtual network embedding with coordinated node and link mapping*. Unpublished Manuscript, 2009.

[17] Csorba, M.J., H. Meling, and P.E. Heegaard: *Ant system for service deployment in private and public clouds*. In *Proceeding of the 2nd workshop on Bio-inspired algorithms for distributed systems, ACM*, pages 19–28, 2010.

[18] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk Graziano Obertelli Sunil Soman Lamia Youseff Dmitrii Zagorodnov: *The eucalyptus open-source cloud-computing system*. `http://open.eucalyptus.com/documents/ccgrid2009.pdf`.

[19] Enokido, T., A. Aikebaier, M. Takizawa, and S.M. Deen: *Energy-efficient server selection algorithms for network applications*. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pages 159–166, November.

[20] Enokido, T., A. Aikebaier, M. Takizawa, and S.M. Deen: *Power consumption-based server selection algorithms for communication-based systems*. In *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, pages 201–208, September.

[21] Feamster, N., L. Gao, and J. Rexford: *How to lease the internet in your spare time*. In *SIGCOMM Comp. Commun. Revi., vol. 37, no. 1*, pages 61–64, 2007.

[22] Feng, Wu chi Feng Wu-chang: *On the geographic distribution of on-line game servers and players*. In *NetGames '03 Proceedings of the 2nd workshop on Network and system support for games*, 2003.

[23] Foundation, Wireshark: *Wireshark*. `http://www.wireshark.org/`.

[24] GARR, Consortium: *Federica*. `http://www.fp7-federica.eu`.

[25] GENI: *GENI - Global Environment for Network Innovations*. `http://www.geni.net/`.

[26] Genovese, Salvatore: *Orange announces complete cloud computing services*, December 2009. `http://cloudcomputing.sys-con.com/node/1226159`.

[27] GridARS: *Gridars*. `http://www.glif.is/meetings/2010/tech/tomohiro-gridars.pdf/`.

[28] Hachimi, Mohamed EL, Marc André Breton, and Maria Bennani: *Efficient QoS implementation for MPLS VPN*. In *22nd International Conference on Advanced Information Networking and Applications - Workshops*, pages 259–263, 2008.

[29] Houidi, I., W. Louati, and D. Zeghlache: *A Distributed Virtual Network Mapping Algorithm*. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 5634–5640, 2008. `http://dx.doi.org/10.1109/ICC.2008.1056`.

[30] Institute, International Modern Media: *Icelandic modern media initiative*. `http://immi.is/Icelandic_Modern_Media_Initiative`.

[31] Jiang, Wenjie, Rui Zhang-shen, Jennifer Rexford, and Mung Chiang: *Cooperative content distribution and traffic engineering in an isp network*.

[32] Li, Lan and Min Tang: *Novel spectral method for server placement in cdns*. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, pages V6–197–V6–199, August.

[33] Lischka, Jens and Holger Karl: *A virtual network mapping algorithm based on subgraph isomorphism detection*. In *VISA '09: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 81–88, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-595-6.

[34] Lu, Jing and Jonathan Turner: *Efficient mapping of virtual networks onto a shared substrate*. Technical report, Washington University in St. Louis, 2006. `http://www.arl.wustl.edu/~{}jl1/research/tech_report_2006.pdf`.

[35] M. Armbrust, A. Fox, R. Griffith A. Joseph R. Katz A. Konwinski G. Lee D. Patterson A. Rabkin I. Stoica and M. Zaharia: *A view of cloud computing*. In *Communications of the ACM 53 (4)*, 2010.

[36] Mathews, J.: *The Future of Virtualization, Cloud Computing and Green IT - Global Technologies & Markets Outlook – 2011-2016*. Market Intel Group LLC, October 2010.

[37] McFedries, Paul: *The Cloud Is The Computer*, August 2008. `http://spectrum.ieee.org/computing/hardware/the-cloud-is-the-computer`.

[38] Mell, Peter and Timothy Grance: *The nist definition of cloud computing (draft)*. `http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf`.

[39] Melo, Marcio, Jorge Carapinha, Susana Sargento, Luis Torres, Tran Phuong Nga, Ulrich Killat, and Andreas Timm-Giel: *Virtual network mapping - an optimization problem*. In *MONAMI - 3rd International ICST Conference on Mobile Networks & Management*, 2011.

[40] Nogueira, J., M. Melo, J. Carapinha, and S. Sargento: *Network virtualization system suite: Experimental network virtualization platform*. In *TridentCom 2011, 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, April 2011.

[41] Nogueira, J., M. Melo, J. Carapinha, and S. Sargento: *Virtual network mapping into heterogeneous substrate networks*. In *IEEE Symposium on Computers and Communications (ISCC) 2011*, June 2011. `http://www.av.it.pt/mmelo/nogueira2011mapping.pdf`.

[42] Nogueira, João: *Demonstraçao de criação de redes virtuais no Âmbito do operador*. Master's thesis, Universidade de Aveiro.

[43] NouVeau: *Nouveau*. `http://netlab.cs.uwaterloo.ca/virtual/`.

[44] OpenNebula.org: *Opennebula.org project*. `http://opennebula.org/`.

[45] Orange Business Services: *Orange business services*. `http://www.orange-business.com/`.

[46] Osana, Y. and S. i. Kuribayashi: *Enhanced fair joint multiple resource allocation method in all-ip networks*. In *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, pages 163–168, April.

[47] PlanetLab: *PlanetLab - An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services*. http://www.planet-lab.org/.

[48] Popek, Gerald J. and Robert P. Goldberg: *Formal requirements for virtualizable third generation architectures*. Commun. ACM, 17(7):412–421, 1974, ISSN 0001-0782.

[49] Portugal Telecom: *Portugal Telecom*. http://www.telecom.pt/.

[50] Portugal Telecom: *Smartcloudpt*. http://www.smartcloudpt.pt/.

[51] Primet, Pascale Vicat Blanc: *Hipernet*. http://www.ens-lyon.fr/LIP/RESO/Software/hipernet/index.html.

[52] Project, OpenStack: *Openstack cloud software*. http://openstack.org/.

[53] Reed, Brad: *Verizon, ibm team on cloud-based storage*, 2010. http://www.networkworld.com/news/2010/033110-verizon-ibm-cloud-storage.html.

[54] Reservoir: *Reservoir*. http://www.reservoir-fp7.eu/.

[55] Roy, N., J.S. Kinnebrew, N. Shankaran, G. Biswas, and D.C. Schmidt: *Toward effective multi-capacity resource allocation in distributed real-time and embedded systems*. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 124–128, May.

[56] Silva, S. da, Y. Yemini, and D. Florissi: *The netscript a c t i v e n e t w o r k s y s t e m*. In *I E E E J S A C, v o l . 1 9 , n o . 3*, pages 538–551, 2001.

[57] Slate, Clean: *Clean slate*. http://cleanslate.stanford.edu/.

[58] SOURCEFORGE.NET: *Iperf*. http://iperf.sourceforge.net/.

[59] Sundararaj, A. and P. Dinda: *Towards vi r tual net - works for virtual machine grid computing*. In *Proc. 3rd USENIX Virtual Machine Research Tech. Symp.*

[60] Touch, J.: *Dynamic internet overlay deployment and management using the x-bone*. In *Comp. Networks, vol. 36, no. 2–3*, pages 1117–135, 2001.

[61] Trilogy: *Trilogy*. http://www.trilogy-project.org/.

[62] UCLP: *Uclp*. http://uclp.ca/.

[63] Weinman, Joe: *The Cloud in the Enterprise: Big Switch or Little Niche?*, April 2010. http://gigaom.com/2010/04/18/the-cloud-in-the-enterprise-big-switch-or-little-niche/.

[64] XenSource, Inc: *Xen*. http://www.xen.org/.

[65] Zhu, Y. and M. Ammar: *Algorithms for assigning substrate network resources to virtual network components*. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.